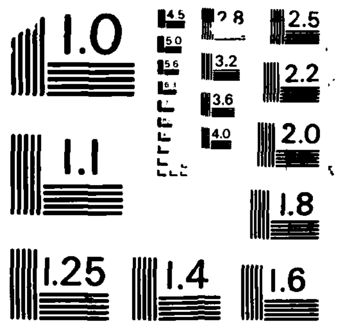
AD-A189 063 HARDWARE FAULT INSERTION AND INSTRUMENTATION SYSTEM: EXPERIMENTATION AND RESULTS(U) LOCKHEED-GEORGIA CO MARIETTA J W BENSON ET AL. MAR 87 DOT/FAA/CT-86/34 1/1

UNCLASSIFIED NAS2-11853 F/G 1/4 NL



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

(2)

DOT/FAA/CT-86/34

FAA TECHNICAL CENTER
Atlantic City International Airport
N.J. 08405

Hardware Fault Insertion and Instrumentation System: Experimentation and Results

J.W. Benson
D.B. Mulcare
W.E. Larsen

March 1987

US Department of Transportation
Federal Aviation Administration

1. Report No. DOT/FAA/CT-86/34	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Hardware Fault Insertion and Instrumentation System: Experimentation and Results		5. Report Date
		6. Performing Organization Code February 1987
7. Author(s) J. W. Benson, D. B. Mulcare, W. E. Larsen		8. Performing Organization Report No. DOT/FAA/CT-86/34
9. Performing Organization Name and Address Lockheed-Georgia Company Marietta, Georgia 30063		10. Work Unit No. NAS2-11853
		11. Contract or Grant No.
12. Sponsoring Agency Name and Address U.S. Department of Transportation Federal Aviation Administration Technical Center Atlantic City Airport, New Jersey 08405		13. Type of Report and Period Covered Contractor Report
		14. Sponsoring Agency Code

15. Supplementary Notes

Point of Contact: W. E. Larsen/MS 210-2
Ames Research Center
Moffett Field, CA 94035

16. Abstract

This report describes the motivation, conduct, and analysis of some 2500 low-level hardware fault cases applied in automated testing at the NASA Ames Reconfigurable Digital Flight Control System (RDFCS) Facility. Fault detection was correlated with hardware and software fault monitoring, and in limited cases, with sensitivity to flight program execution modes. Results obtained have been statistically assessed to ascertain system-level reliability implications based on a model that is described herein. The overall methodology/facility itself has been critiqued and found to constitute a promising enhancement to current practice.

17. Key Words (Suggested by Author(s)) Automated Testing, Digital Flight Controls, Fault Coverage, Fault Latency, Fault Modeling, Hardware Faults, Monitors, Statistical Fault Data	18. Distribution Statement Unlimited Subject Category 38

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages	22. Price*

*For sale by the National Technical Information Service, Springfield, Virginia 22161

FOREWORD

This report describes the theory, conduct, and results of automated low-level hardware fault testing that was performed as part of an FAA-sponsored program entitled "Methods for the Verification and Validation of Digital Flight Control Systems." Specifically, this work was accomplished as Subtask 4.5.7 of Contract NAS2-11853, Modification 2. The intent has been to demonstrate, describe, and critique the intensive, statistically based assessment of fault detection and recovery mechanisms, especially as impacts system reliability. Such an approach is considered to offer exceptional new capability for establishing airworthiness of critical digital flight systems.

Over 2500 simulated hardware fault cases were applied at the chip pin level in the Reconfigurable Digital Flight Control System (RDFCS) Laboratory at NASA Ames. Most testing was done on an open-loop basis; the more persistant faults, however, were subjected to closed loop simulator testing in order to involve explicit fault detection mechanisms. The test results are in reasonable accord with similar prior data, conform well with the associated fault models, and indicate the value of this type testing for practical system validation efforts.

Acknowledgement

The authors would like to acknowledge the contribution of John McGough of Allied/Bendix Aerospace, Flight Systems Division, for his advice and assistance in the planning and interpretation of the experiments. It is fair to say that the contents of this report are due as much to his efforts as to ours.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

LIST OF FIGURES

LIST OF FIGURES (CONT'D)

LIST OF TABLES

1.0 INTRODUCTION

The Federal Aviation Administration (FAA) is executing a multi-faceted plan to generate and disseminate information deemed vital in establishing the airworthiness of near-tearm civil transports. Particular emphasis is directed toward the integrated application of state-of-the-art tools, techniques, methodologies, criteria, and data to evaluate the integrity, reliability, and capability of the onboard software-based digital flight control and avionic systems (Reference 1). This report summarizes the activities of one of the FAA-sponsored tasks. As such, it will later be incorporated partially in Volume II of their Digital System Validation Guidebook and presented at an associated Government/Industry Workshop during 1988.

The FAA's research agenda in digital systems assurance technology has been motivated by the trend within the aviation community toward more highly integrated fault-tolerant digital architectures. The criticality of these complex systems, moreover, is compounded by the range of their application in varied functions such as structures, propulsion, electrical power, flight control, and avionics. Fortunately, the assurance technology being addressed is quite generic, so the focus here on digital flight controls is representative but not limiting as far as the applicability of results is concerned.

This particular task is the third in a sequence of FAA-sponsored program elements focusing on low-level hardware fault detection. Reference 2 detailed a study setting forth options for modifying the RDFCS Facility to support such experimentation, and Reference 3 documented the mechanization of certain modifications actually needed to perform the work reported herein. Note that these modifications, which are described in Section 5, extended the utility of a Fault Injector System (FIS) developed by the Charles Stark Draper Laboratories (CSDL) (Reference 4). The added capability included automated test case sequence application, instrumentation, timing, and interpretation centered upon the FIS. Hence, the expanded test system has been referred to as the Fault Insertion and Instrumentation System (FIIS).

THIS PAGE LEFT INTENTIONALLY BLANK

2.0 EXECUTIVE SUMMARY

Fault injection experiments were conducted using the Fault Injection and Instrumentation System (FIIS), which was developed, in a previous program, by the Lockheed-Georgia Company under contract to the FAA. The facility was resident at NASA Ames Research Center at Moffett Field, California. The target hardware was a dual flight control computer, which was part of a dual/dual flight control system for the Lockheed L-1011 Tristar. Each dual lane featured a Rockwell/Collins CAPS-6 digital computer as the principal computational element. 2715 distinct stuck-at and invert faults were injected on the pins of devices of the data path and control cards of the CAPS-6 while executing inner loops and autopilot modes. Faults were detected by hardware comparators located in the secondary actuator drive electronics or by self-test.

2.1 PREVIOUS FAULT INJECTION EXPERIMENTS

Similar fault injection experiments were conducted previously, and independently, by Bendix (Reference 5) and the (CSDL) (Reference 4). In the Bendix study a Bendix BDX-930 flight control computer was simulated in software at the gate-level and faults were injected at gate nodes. In the CSDL study faults were injected on pins of digital devices of FTMP (Fault-Tolerant Multi-processor) (References 4, 6, 7), which also featured the CAPS-6 as the principal computing element. The fault injector hardware was identical to that used in the present study. The methodology and results of these studies were examined (Section 4) in order to establish guidelines for the FIIS experiments, to avoid a repetition of results and to identify gaps and ommissions which could be remedied by further experimentation. In addition, a number of unresolved issues associated with flight control system (FCS) survivability assessment were identified (Section 4.2).

2.2 OBJECTIVES

The objectives of the FIIS experiments were:

1) Corroborate and augment, if possible, the results of the Bendix and CSDL studies.

2) Provide a database of detection coverage and fault latency which future experimenters could use as goals or as a basis for comparison.

3) Evaluate the FIIS fault injection methodology. It was hoped that this methodology would be the first step in establishing guide-lines for future fault injection experiments and fault detection coverage estimation.

4) Provide a database for the construction of single and multiple-fault models which could be used by reliability programs to assess FCS survivability.

5) Identify unresolved issues associated with FCS survivability assessment and recommend studies to resolve them.

2.3 RESULTS AND CONCLUSIONS

A detailed description of the results of the FIIS experiments is given in Section 7, with summary and conclusions, in Section 9. The results and conclusions are briefly summarized here.

1) Most detected faults are detected by comparators within 500 ms of their occurrence.

2) Most faults are activated by operational software or computer hardware and not by sensor inputs.

3) Most faults (96.3%) were detected while executing the baseline program, i.e., inner loops and altitude hold; 1.3% were detected while executing an autopilot mode; 2.4% were not detected by comparators at all. The undetected faults could accumulate in redundant channels and eventually be activated by an infrequently used outer loop computation.

4) All undetected faults were detected by self-test. Preflight or background self-test is an effective way to eliminate latent faults.

5) Faults which are activated by the baseline program produce errors within 500 ms of their occurrence; faults activated by the autopilot modes (and not by the baseline) produce errors over somewhat longer intervals, e.g., up to 6.5 sec.).

6) 6.11% of all faults were "don't care" faults. These were subsequently identified and their effects removed from the tabulations.

7) FIIS test results generally corroborated the results of the Bendix and CSDL studies. This was surprising considering the dissimilarity of sensor input selection, computer architecture and operational software. Similarity at the lower levels of computer hardware implementation may explain this phenomenon.

8) The FIIS methodology was thoroughly tested during the study. FIIS performance was impressive, especially in the areas of (a) fault generation capability, (b) ease of use, (c) fidelity, and (d) the real time nature of the experiments. This latter feature made it possible to make runs of 15 or more seconds of real time, which would have been impracticable with a software simulator. This high productivity during fault testing facilitates comprehensive, and hence more conclusive, flight computer hardware testing.

2.4 RECOMMENDED FUTURE STUDIES

A number of issues associated with FCS survivability assessment remain unresolved. To resolve some of these issues the following studies are recommended:

1) Failure Modes of Digital Devices

   As a practical first step, Bendix and CSDL injected stuck-at faults on pins or gate nodes. It remains to be seen to what extent these faults represent failure modes of real devices. Hence, the deceptively difficult task of determing actual low-level hardware failure modes is required.

2) Pin-Level Versus Gate-Level Faults

   It is not likely that real failure mode data will become available in the near-term. In the interim, experimenters will continue to use stuck-at faults. The Bendix study indicated a significant difference between pin-level and gate-level detection coverage. At issue is the validity of either fault type and the extent to which one is more or less latent than the other.

3) Single and Multiple-Fault Models

   These models are essential ingredients of reliability assessment programs. Simple models for permanent faults were proposed in Section 4. These models should be improved and extended to include intermittent/transient faults, for these tend to occur relatively often and they may substantially impact system reliability.

4) Reliability/Survivability Assessment of an FCS

   A reliability parameter sensitivity study should be conducted to (a) identify critical ranges of single and multiple-fault model parameters and (b) determine inflight, preflight and maintenance test coverage requirements to reduce latent faults to acceptable levels. These issues need to be addressed so that more confidence can be placed in system reliability predictions.

THIS PAGE LEFT INTENTIONALLY BLANK

3.0 RESULTS OF PREVIOUS FAULT INJECTION EXPERIMENTS

Prior to the FIIS experiments, most of the published information on data latency in digital flight control systems was obtained from two important studies conducted, independently, by Bendix (Reference 5) and CSDL (Reference 4). The methodology and results of these experiments were examined in detail in order to establish guidelines for the FIIS experiments, to avoid a repetition of results and to identify gaps and ommissions which could be remedied by further experimentation. Here the intent is to demonstrate an effective methodology for calibrating fault detection/latency in a dependable manner that relates readily to system-level reliability.

3.1 OVERVIEW OF THE BENDIX AND CSDL STUDIES

Bendix Study

The purpose of this study was to estimate (1) detection coverage of several candidate self test programs and (2) fault latency in a conventional, redundant digital flight control system. Fault injection experiments were conducted on a software simulated version of the Bendix BDX-930, bit-sliced flight control computer, which featured the 2901A arithmetic logic unit (ALU) as the principal processing element. The simulated components consisted of the central processor unit (CPU) and micromemory, program memory and random access memory (RAM) scratchpad. The I/O circuitry was not simulated: sensor inputs were randomly generated and deposited in scratchpad memory at the start of each computational frame. Each device of the CPU was represented by a gate-equivalent circuit and stuck-at faults were injected at the gate nodes, something not possible in actual hardware fault testing. The micromemory was represented functionally and faults were simulated by reversing the logic states of single bits.

The program and RAM memories were also represented functionally but no faults were injected into these devices. Faults were randomly selected and injected, one at a time. Fault selection was weighted in proportion to the failure rate of the device, i.e., the average number of faults injected into a particular device was proportional to the device failure rate. The simulation technique was "parallel mode" (Reference 8), which allowed for the simultaneous simulation of up to 32 computers, one of which was always the non-failed version. This made it possible to compare the responses of the failed and non-failed computers at any time in the compute cycle and at any device.

Although all devices were simulated at the gate-level, faults were injected at both the pin-level and gate-level in order to determine differences in detection coverage between the two fault types.

Two types of experiments were conducted, depending upon the method of detection:

1) To determine the effectiveness of comparison monitoring, the computed outputs of a failed and non-failed computer were compared at the end of each frame. Any discrepancy was defined as a "detected fault". These comparisons were performed by the simulator executive and did not involve the detection mechanisms that would normally be resident in the flight control computers. Each computer executed the same flight control program, which consisted of the inner loops for a high performance aircraft. In order to reduce simulation time, a fault run was terminated immediately after detection or after eight repetitions, whichever occurred first. To avoid any ambiguity in the comparison process, each computer received identical sensor inputs at the start of each frame. The simulation was conducted "open-loop," with sensor values selected independently and at random.

2) To determine the effectiveness of self-test, each fault was injected and a candidate self-test program was executed. A fault was defined as "detected" if the mechanism of the self-test routine so indicated.

CSDL Study

The purpose of this study was to assess the fault detection, identification and reconfiguration capabilities of the CSDL-designed FTMP (References 4, 6, 7). This system featured the Rockwell/Collins CAPS-6 computer as the principal processing element. The CAPS-6 is similar to the BDX-930 in that it, too, is a bit-sliced processor utilizing the 2901A, arithmetic logic unit (ALU). FTMP is a triple modular redundant (TMR), bit-synchronized multiprocessor designed for ultra high reliability and fault-tolerance. The design is based on independent processor-cache memory modules and common memory modules (Reference 6) which communicate via redundant serial busses. All information processing and transmission is conducted in triplicate. Data transmitted over the bus network is monitored by Bus Guardian Units (BGUs), which compare the transmitted data, bit by bit. Faults are detected when they produce errors at the BGUs. Faults are also detected by self test programs, which are executed continuously, in the background

The fault injection experiments were conducted on real hardware and stuck-at faults were injected, one at a time, on device pins of one of the three processing elements. (The fault injection hardware was identical to that used in the FIIS experiments.) Faults were systematically selected (unweighted) and injected on pins of eight circuit boards: CPU Data Path, CPU Control Path, Processor Read Only Memory, Processor Cache Controller, Bus Guardian Unit, Transmit Bus Interface, Poll and Clock Bus Interfaces, and System Bus Controller. FTMP executed a flight control software program consisting of inner loops and autopilot modes, which were patterned after the L-1011 Tristar flight control system. Fault runs were terminated after 15 seconds of real time. Unlike the Bendix experiments, the simulation was "closed-loop," i.e., the sensor values were obtained from the motion parameters of a simulated aircraft. For the more persistent types of faults, closed-loop testing may introduce added fault detection capability via runstream/datastream fault sensitivity.

3.2 RESULTS

1) Both studies indicated that most faults were detected within a few computational frames of their occurrence (e.g., 0-500ms). The fact that both studies produced this result was surprising considering the dissimilarity of input sensor selection, computer architecture and operational software. Because fault runs were terminated after eight repetitions, the Bendix study gave no information about detection beyond 800ms (assuming 100 ms/frame). The CSDL results indicated that relatively few faults (2%-4%) were detected in later repetitions and these were detected by background self-test programs (Reference 4). From these results it was conjectured that most faults are activated by the baseline software program and independently of the input sequence. This phenomenon is generally accompanied by shorter latency times.

2) CSDL injected 21,055 faults (each pin fault was injected 5 times, at different locations of the program) and 3,637 (17.3%) were undetected. CSDL estimated that about 3000 of the undetected faults were "don't care" faults, i.e., faults on unused pins or on signals that were always low or always high under normal circumstances. Of the remainder, a few were analyzed and found to be "don't cares". In the Bendix study, 3000 faults were injected during the self-test evaluation phase and each undetected fault was analyzed. It was found that 466 faults (15.5%) were "don't cares" and, of these, 245 occurred in the bits of the micromemory and 47 in the control PROMS. When memory bit-faults were excluded, 7.2% of gate-level faults were "don't cares" and 6% of pin-level faults were "don't cares". Both Bendix and CSDL concluded that the identification of "don't care" faults was a non-trivial task, but essential to excluding irrelevant test results as in the form of the completely undetected faults.

3) The Bendix study found that 86.5% of gate-level faults (excluding bit-faults in the micromemory and control PROMS) were detected during the eight repetitions of the FCS program, based on 148 injected "care" faults. Pin-level faults were not injected while executing the FCS program. The equivalent coverage in the CSDL study is not clear because of the large number of undetected, unanalyzed faults, most of which (but, possibly, not all) were "don't cares." CSDL did conclude,

however, that between 2% and 4% of all detected faults were detected by the background self-test programs and not by the comparators. Since self-test was not executed during the FCS experiments, Bendix would have concluded that these faults were undetected.

4) The Bendix study showed that 88% of 2901A, gate-level faults were detected during the FCS experiments, based on 52 injected "care" faults. The CSDL study gave no results for this device.

5) In the Bendix study, 97.4% of gate-level faults and 97.6% of pin-level faults were detected by self-test, based on 2234 and 376 injected "care" faults, respectively. The gate-level coverage excluded bit-faults in the memory elements. When these faults were included, coverage was 92%. The self-test program consisted of 346 assembly language instructions and required 2ms-3ms to complete. The CSDL study gave no results for self-test coverage.

6) The Bendix study concluded that gate-level faults were more difficult to detect than pin-level faults, especially when faults were injected into single bits of the memories. When executing software, other than the FCS, the ratio of gate-level to pin-level undetected faults was a factor of two. Unfortunately, "don't care" faults were not identified in these runs. As a result, detection coverage estimates tended to be pessimistic.

3.3 CONCLUSIONS

The principal conclusions of the Bendix and CSDL studies are:

1) Most detected faults are detected within a few computational frames of their occurrence.

2) The proportion of faults not detected by comparison monitoring while executing the baseline program can range from 2% to 13.5%.

3) Although not discussed in the Overview, CSDL results showed that detection, isolation and recovery could take up to several seconds, during which time the system is potentially vulnerable to second faults. The occurrence of a second fault, before the first fault is isolated, could confuse the majority vote and result in a system breakdown.

4) Both studies indicated a significant proportion of "don't care" faults (between 6% and 17.3%). These faults are difficult to identify. Unidentified "don't care" faults could result in uncertain and pessimistic detection coverage estimates.

5) Self-test coverage of 95% is easily achieved for pin-level faults and for gate-level faults, if memory bit-faults are excluded.

6) There is a significant difference in detection coverage between pin-level and gate-level faults particularly if memory bit-faults are included in the latter.

The Bendix and CSDL studies were intended to provide an initial database for eventual FCS survivability assessment. In the next section some of the unresolved issues connected with this assessment are discussed.

THIS PAGE LEFT INTENTIONALLY BLANK

4.0 UNRESOLVED ISSUES OF FCS SURVIVABILITY ASSESSMENT

Prior to conducting the FIIS experiments, a survey was made to determine some of the unresolved issues connected with FCS survivability assessment. The purpose of the survey was to identify issues which could be resolved by further experimentation.

In order to avoid confusion and ambiguity later on, we give a few informal definitions of fault types and associated processes, some of which were suggested by previous fault injection experiments and others, from a survey of the literature.

4.1 DEFINITIONS: FAULT TYPES AND ASSOCIATED PROCESSES

Malfunction:	An error at the output of a digital device.
Fault:	An internal condition of a device which causes a malfunction for some combination of input and internal state.
Excitation:	A condition such as an input or internal state which, in conjunction with a fault, causes the device to malfunction.
Permanent Fault:	A fault which persists indefinitely.
Intermittent Fault:	A fault which occurs and reoccurs, intermittently.
Healed Fault:	A former fault which can no longer produce a malfunction for any excitation.
Transient Malfunction:	A malfunction which occurs for a brief period of time and then disappears. Sometimes referred to as a "transient fault."
Latent Fault:	A fault which has not yet produced a malfunction.
Stuck-at Fault:	A fault which causes a gate node to assume a logic 0 or a logic 1 state, denoted by S-a-0 and S-a-1, respectively.
Pin-Level Fault:	A stuck-at fault on an input or output pin of a digital device.
Gate-Level Fault:	A stuck-at fault on a gate node of a gate-equivalent circuit.

Don't Care Fault:	A fault which cannot produce a malfunction for any combination of input and internal state.
Correlated Faults:	Two or more faults which can produce a malfunction for the same excitation.
Near-Coincident Fault:	A fault which occurs before a prior fault has been detected and isolated.
Baseline Program:	A set of software that is executed continuously, e.g., an inner-loop.
Auxiliary Program:	A software program that is executed occasionally, e.g., an outer-loop.
Active Fault:	A fault which can produce a malfunction while executing the current on-line program.
Benign Fault:	An inactive fault wrt the current on-line program but active wrt to (with respect to) some other program.
Alpha-Fault:	An active fault wrt the baseline program.
Beta-Fault:	A benign fault wrt the baseline program but active wrt an auxiliary program.

4.2 UNRESOLVED ISSUES

1) Effects of Latent Faults

Latent faults may accumulate in different lanes of an FCS, particularly if pre-flight self-testing is limited in detection coverage. Their eventual activation by a single excitation event (e.g., the execution of a seldom-used outer-loop) could result in a rapid loss of lanes, even if successively detected. Such an event could have a significant impact on system survivability. Required for this assessment:

- o An understanding of the mechanisms that transform latent faults into error-producing faults.
- o Estimates of the accumulation of latent faults.
- o Extent to which latent faults are correlated across lanes.
- o Single and multiple-fault models incorporating latent faults.
- o Reliability analyses.

The Bendix and CSDL studies showed that the build-up of latent faults could range fron 2% to 13.5%, especially if background self-test was not employed. Neither study examined the potential correlation of these faults.

2) Effects of Intermittent/Transient Faults

These faults affect the rate at which errors are produced. (Reference 9) addresses their affect on FTMP survivability. These results should be extended to include a wider class of FCS architectures. Transient faults could result in a premature disengagement of FCS lanes if their rate of occurrence greatly exceeds that of conventional failures. Most flight control systems avoid premature, permanent disengagement by allowing the reengagement of a previously failed lane under certain conditions. Required for this assessment:

- o Occurrence, duration and reoccurrence rates of intermittent/transient faults.
- o Relative proportion of intermittent/transient faults versus permanent faults.
- o Single and multiple-fault models incorporating intermittent/transient faults.
- o Reliability analyses.

3) Effects of Near-Coincident and Multiple Faults

(Reference 10) addresses the potential problem of near-coincident faults in a multiprocessor system. The Bendix and CSDL studies gave an indication of the expected "time-on-risk" during which the FCS is vulnerable to second faults. Multiple fault conditions have yet to be evaluated, e.g., the effect of a latent fault followed by an active fault, particularly when the latent fault may preclude detection of the active fault. Required for this assessment:

- o Fault injection experiments.
- o Multiple-fault models.
- o Reliability analyses.

4) Proportion of Alpha/Beta Faults

The Bendix results were obtained while executing the baseline FCS program. Thus, all detected faults were alpha-faults. It was never determined if any of the undetected faults would have been detected while executing an auxiliary program. If so, then the execution of an infrequently-used outer-loop could simultaneously activate latent faults in different lanes. The CSDL study gave no information on this subject. Required for this assessment:

- o The relative proportion of alpha and beta-faults.

5) Dynamics of Alpha/Beta Faults

The Bendix study found that alpha-faults produced errors, at comparators, within several frames of their occurrence. Beta-faults, once activated by an auxiliary program, could, conceivably, produce errors at a different rate or be more difficult to detect by self-test. Required for this assessment:

- o Fault injection experiments which identify individual faults and their detection while executing inner and outer-loops.

- o Estimates of self-test coverage for both types of faults.

6) I/O Faults

Detection coverage estimates of I/O hardware faults are not available, probably because of the difficulties in simulation. Required for this assessment:

- o Failure mode data for I/O devices.

- o Techniques for simulating I/O hardware and associated faults.

- o Fault injection experiments on I/O hardware.

7) Failure Modes of Digital Devices; Pin-Level Versus Gate-Level Fault Models

The Bendix study indicated a significant difference between pin-level and gate-level fault detection. At issue is the validity of either model. Required for this assessment:

- o Failure mode data of actual digital devices.

- o An analysis of the relative latency of pin-level and gate-level faults.

8) Weighted Versus Unweighted Faults

In the Bendix study, faults were weighted according to the failure rate of the device. In the CSDL study all faults were treated as equally probable. At issue are a) the validity of either approach and b) the relative, resultant differences in detection coverage. Required for this assessment:

- o Establish, once and for all, the definition of "fault detection coverage."
- o Estimates of detection coverage for both approaches.

9) Guidelines for Simulation Testing

In order to obtain a industry-wide uniformity in simulation testing it is necessary to establish guidelines. The guidelines should include fault models, apportionment of fault types, methods of simulation, FCS scenarios, self-test coverage validation procedures, etc. Required for this assessment:

- o Failure mode data.
- o Recommended fault models.
- o Weighted versus unweighted faults
- o Recommended methods of simulation.
- o Effects of latent faults on survivability.
- o Establishment of confidence levels and accuracy requirements of test results.
- o Recommended FCS scenarios.
- o Establishment of single and multiple-fault models and the identification of critical model parameters.

10) Self-Test Requirements

A typical FCS incorporates several self-test programs for inflight, preflight and maintenance testing. At issue are the coverage requirements of each test and the rationale for the requirements. Required for this assessment:

- o Build-up of latent faults and the effect on survivability.
- o The determination of an acceptable rate of build-up via reliability analysis.
- o Effects of periodic repair, possibly requiring higher coverage levels.

11) Proportion of Faults Affecting Surface Commands

Required for this assessment:

- o Fault injection experiments.

12) Proportion of Faults Affecting the Monitoring Process

Required for this assessment:

- o Fault injection experiments.
- o Analysis of failure rates of monitoring components.

13) Reliability Assessment Programs

A number of reliability assessment programs are now available to the flight control community, e.g., CARE III, HARP, ARIES, CARSRA, SURE, etc. Evidently, it remains to be seen whether these programs are capable of representing and analyzing the fault combinations identified in this section. Required for this assessment:

- o Establishment of single and multiple-fault models.
- o Evaluation of the capabilities of the candidate programs.

4.3 STRAWMAN SINGLE-FAULT MODEL

In planning the FIIS experiments, it was agreed, by all of the participants, that the principal and fundamental objective was to obtain a database for the reliability assessment of flight control systems. As a consequence, it was necessary to anticipate the kind of data that might be required. Recognizing that the key elements of a reliability assessment program are single and multiple-fault models, it was decided to generate a single-fault model in order to identify transition parameters which could be obtained from the FIIS experiments. The structure of the single-fault model was suggested from the experience gained from the Bendix and CSDL studies. The model is semi-markov and appears to have sufficient degress-of-freedom to model a wide variety of fault and error dynamics, particularly as observed in the Bendix and CSDL fault injection studies. The model is intended to represent permanent faults, only. The mathematically more tractable markovian version is shown in Figure 1. The following parameters describe the fault and error dynamics:

A = ACTIVE α- FAULT
a = ACTIVE β- FAULT
b = BENIGN FAULT

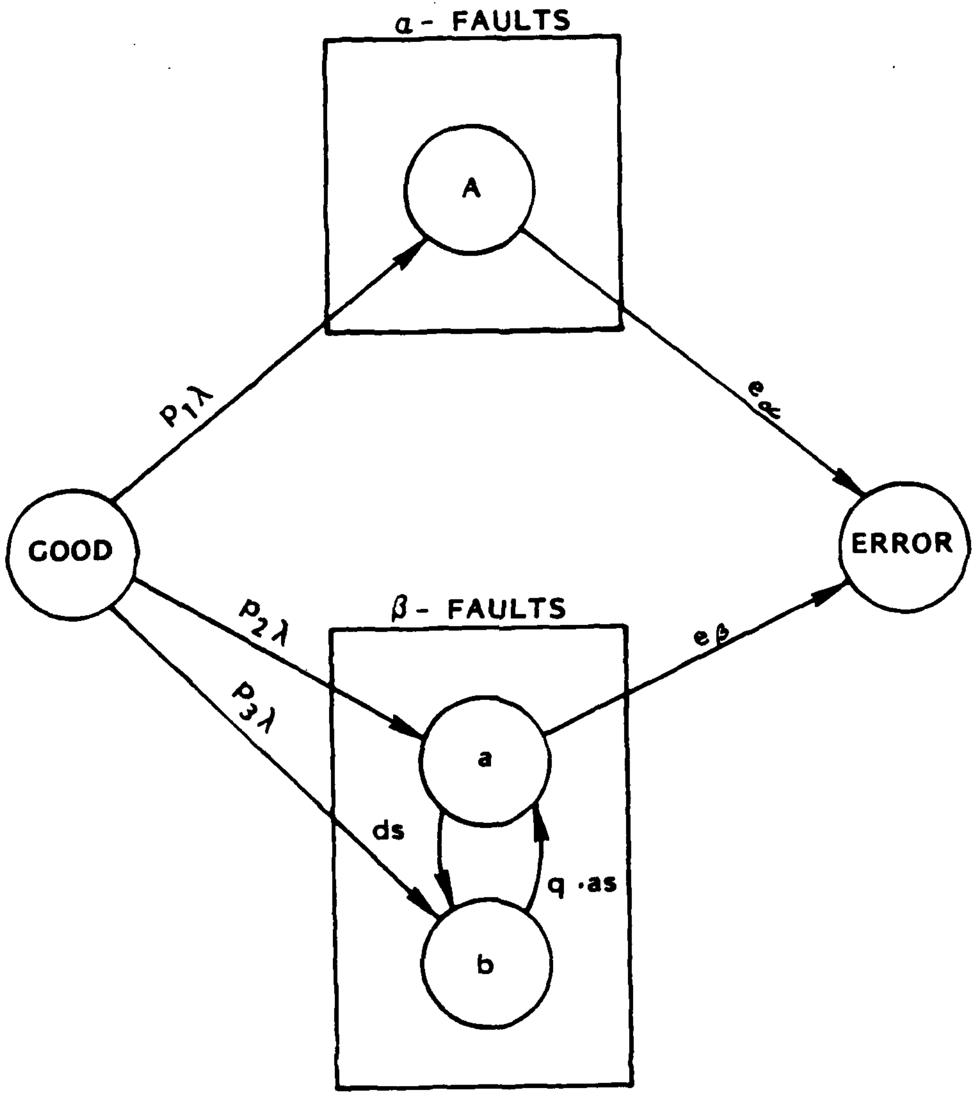


Figure 1. Single-Fault Model Markov Version

λ	=	failure rate of a lane (failures/hour)
p1	=	proportion of alpha-faults
p2	=	proportion of beta-faults which are active at their occurrence
p3	=	proportion of beta-faults which are benign at their occurrence
e_α	=	rate at which alpha-faults produce errors (errors/hour)
e_β	=	rate at which active beta-faults produce errors (errors/hour)

[It was anticipated that alpha and beta-faults produce errors at the same rate. However, the experiments showed a small difference between e_α and e_β.]

as	=	rate at which auxiliary programs are brought on-line (programs/hour)
1/ds	=	average duration of an auxiliary program (hours/program)
q	=	proportion of auxiliary programs that activate a benign fault

We note that

$$p1 + p2 + p3 = 1.$$

The parameters of the single-fault model are not independent. In fact,

$$p1 + p2 + p3 = 1$$

$$p2 = (p2 + p3)q[as/(as+ds)] = (1-p1)q[as/(as+ds)]$$

$$p3 = (p2 + p3)[1 - q(as/(as+ds))] = (1-p1)[1-q(as/(as+ds))].$$

The remaining parameters p1, as, ds and λ are independent.

From the results of the FIIS and previous experiments it was expected to obtain estimates of p1, p3, e and q. The parameters p2, as, and ds would be determined from flight control operational scenarios, via estimates of a.) duration and b.) time between call-ups of auxiliary programs.

4.4 EXTENSION TO MULTIPLE-FAULT MODELS

To illustrate the potential use of the single-fault model in reliability studies, a simple multiple-fault model will be constructed. The model is for a multiprocessor system, such as SIFT or FTMP, and is shown in Figure 2. The model is constructed on the basis of the following groundrules:

1) The system contains a core of powered, triplex processors and an unlimited number of unpowered spares.

2) The core uses identical inputs and employs majority voting of computed outputs to detect and isolate a faulty processor. Single fault detection coverage and isolation is 100%.

3) When a faulty processor is identified it is immediately replaced by a spare, which is then powered.

4) The failure rate of an unpowered spare = 0.

5) It is assumed, conservatively, that active faults in different processors results in loss of control (LOC) even if neither one has produced an error. (This assumption reduces the number of states.)

6) There is only one auxiliary software program, i.e., $q=1$. (This simplifies the transition probabilities.)

This model is interesting because LOC cannot occur due to exhaustion of processors but only as a result of near-coincident active faults or the simultaneous activation of benign faults in different processors.

We illustrate the derivation of transition rates by deriving the rates from state #7. This state depicts the condition that one processor contains an active fault and the other two processors contain benign faults.

Transition from State #7 to State #6

The active fault produces an error while the other faults remain benign. The error is detected and the faulty processor is replaced.

Transition from State #7 to State #9 (LOC)

The auxiliary program is called on-line or one of the two processors, containing benign faults, experiences an active fault.

It is emphasized that this multiple-fault model is presented for illustrative purposes, only. It is recognized that the generation of appropriate multiple-fault models is a difficult undertaking, particularly when correlated faults are included, and is far beyond the scope of this study.

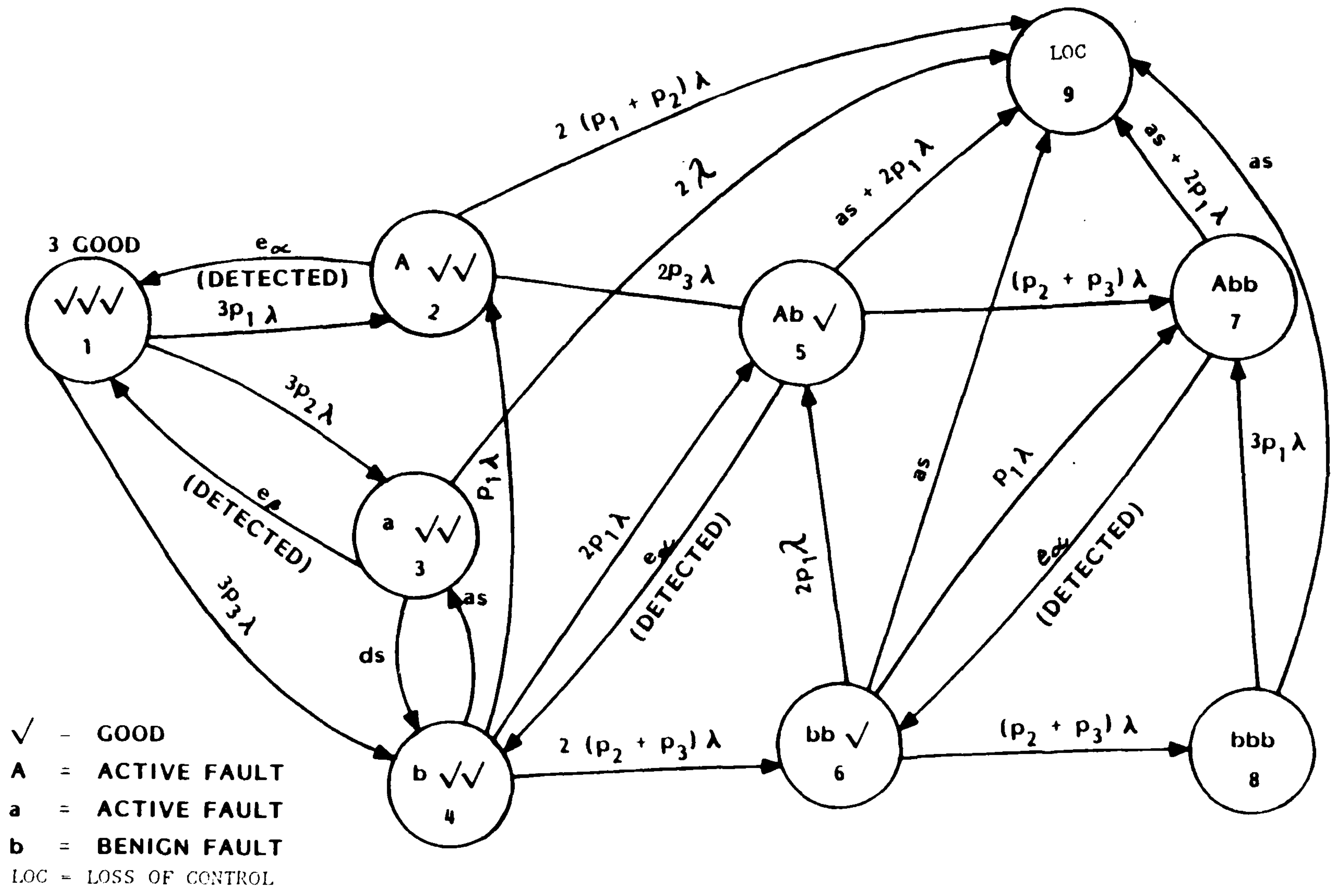


Figure 2. Multiple-Fault Model for a Multiprocessor System

(with a single auxiliary software program)

5.0 DESCRIPTION OF FIIS CAPABILITIES

The following is an abstracted version of certain parts of Reference 3, a companion document to this report. The intent here is to provide adequate information about the FIIS testing capability to enable better insight into the ensuing experimental results. For the type and scope of testing undertaken, it was necessary to extend the capabilities of the RDFCS. In particular, it was necessary to:

- o Automate sequences of command files applied to the Draper's FIS software for test productivity.
- o Add a precise timing mechanism to measure the inverval from test stimuli application to its detection for latency data.
- o Expand the instrumentation capability through the PDP 11/04 utility computer for bandwidth and selectivity.
- o Set up test case post-processing in the PDP 11/60 for interpreting and storing results.
- o Establish a test executive in the PDP 11/60 for the unified control.

The resulting FIIS capability for the RDFCS facility is depicted in Figure 3. Added hardware includes an external clock and an associated DR11C processor interrupt so that fault detection in an FCC is quickly evident to the PDP 11/60 through a DR11C interrupt. The rest of the modifications are implemented in software as described in Reference 3. Considerable additional software has been developed to analyze, reduce, and present the results presented herein.

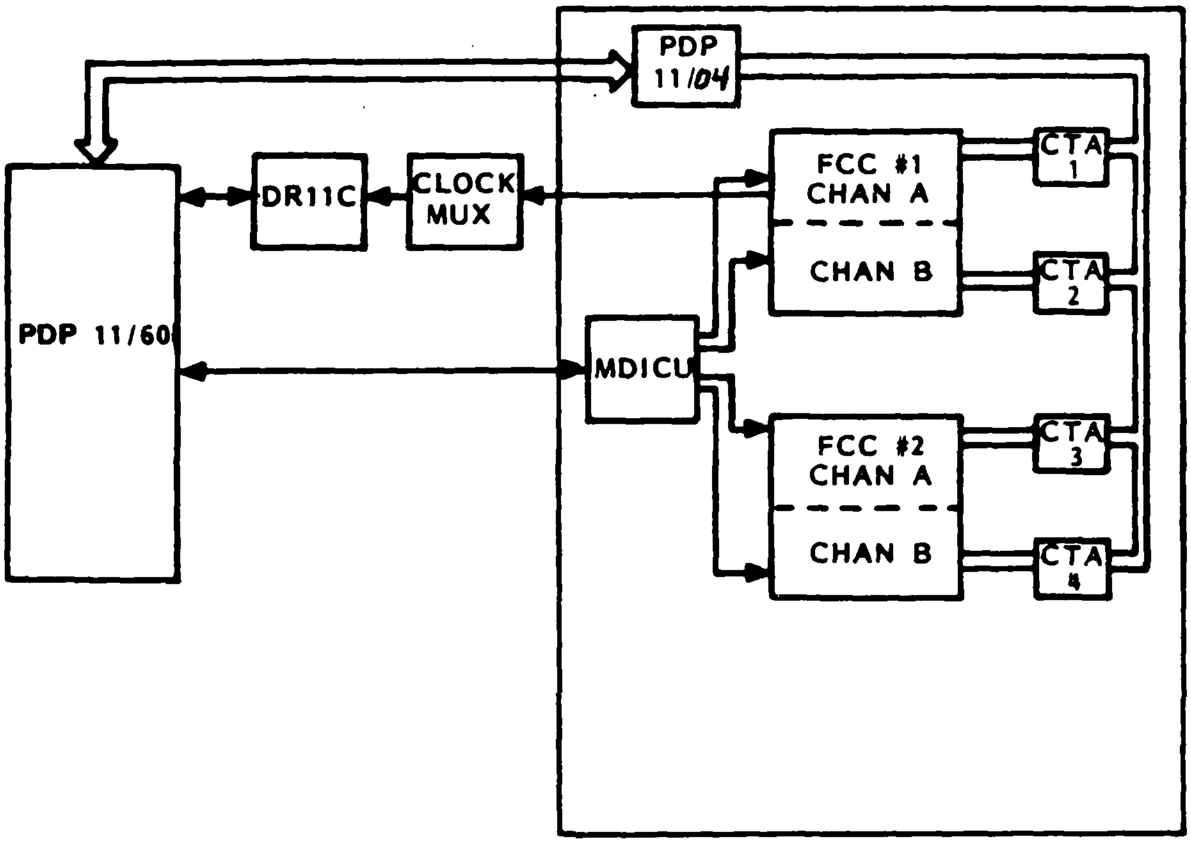


Figure 3. Fault Insertion and Instrumentation System

6.0 OBJECTIVES OF THE FAULT INJECTION EXPERIMENTS

The resolution of the issues of the previous section requires a combination of fault injection experiments, industry surveys and reliability analyses. Issues resolvable by fault injection experiments were analysed to determine what additional data could be obtained by the FIIS experiments. This resulted in the following specific objectives of the fault injection experiments:

1) Obtain estimates of the proportion of alpha and beta-faults, i.e., the proportion of faults detected while executing the inner and outer-loops, respectively.

2) Estimate and compare the dynamics of alpha and beta-faults, e.g., obtain latency histograms for each type of fault.

3) Assess the ability of the self-test program to detect alpha and beta-faults.

4) Estimate the proportion of faults affecting the control surfaces.

5) Estimate the parameters of the strawman single-fault model, i.e., p1, p3, e_α, e_β and q.

6) Compare results for weighted and unweighted faults.

7) Corroborate and compare results with those of Bendix and CSDL.

8) Evaluate the FIIS methodology with respect to its ability to obtain meaningful fault latency data efficiently and cost effectively.

9) Provide a database for future studies. It is intended that the collective results of the Bendix, CSDL and FIIS experiments will be used by future experimenters as goals and for comparison purposes. An important feature of the FIIS experiments is the use of flight-certified FCS as the target computer. It is hoped that future flight control systems will produce similar results for the stuck-at class of faults.

Permanent Versus Intermittent/Transient Faults

The initial intention was to simulate intermittent/transient faults especially since the fault injection hardware provided this capability. However, in planning these experiments, it quickly became apparent that simulating intermittent/transient faults and fault scenarios required the selection of too many arbitrary parameters to yield convincing results. Among these were:

1) duration and reoccurrence rates,

2) place of occurrence within the FCS program,

3) failure modes of intermittent/transient faults.

In addition, the effort required to set-up these experiments would have exceeded the time and resources of the study. Consequently, it was decided to simulate permanent faults, only.

Pin-Level Failure Modes

Because of the extensive fault injection capability of FIIS a large variety of pin-level faults could have been simulated. To mention but a few (see Reference 3 for the complete list):

1) individual input/output pins faulted S-a-0/S-a-1,

2) individual input/output pins faulted by inverting their nominal values ("invert" faults),

3) one or more input/output pins faulted as a function of the values of other input/output pins.

It was recognized that the results (e.g., latency) would be strongly affected by the kinds of failures injected and especially by the proportion of input vectors which were changed by a fault. For example, a S-a-0 on a single pin would change half of the input vectors whereas an invert fault on the same pin would change all of the input vectors. In the absence of failure mode data for the actual devices it was considered too innovative to introduce fault combinations other than S-a-0/S-a-1 of single pins. In addition, it was desirable to employ the same groundrules as the Bendix and CSDL experiments so that the results could be compared. However, as a concession to innovation, it was decided to inject invert faults, with the proviso that the results would be separately tabulated to test the relative latency of stuck-at versus invert faults.

7.0 RESULTS OF FIIS EXPERIMENTS

7.1. TEST GROUNDRULES/PROCEDURES

Overview

The target hardware is a dual FCC, which implements a complete flight control system for a commercial aircraft (Lockheed L-1011 Tristar). Faults were detected by comparison monitoring of computed variables which were periodically exchanged between FCCs. Either FCC could request and affect system disengagement if it observed a discrepancy in any of these monitored variables. System disengagement constituted "detection." In addition, the system featured hardware comparators which were located in the secondary actuators. These comparators, which were not faulted, effectively measured the difference between the surface commands generated by the two FCCs. Again, comparator exceedances constituted "detection" and resulted in system disengagement. The experiments were conducted in open-loop and closed-loop scenarios.

Open-Loop Scenario

The simulated airframe was disconnected for all runs. As a consequence, sensor inputs were invariant. In this configuration the operational program consisted of the inner loops, mode logic servicing, executive functions, synchronization, control panel and display servicing, voting and monitoring and intercomputer communications. The open-loop program executed approximately 11,000 assembly instructions every 50 ms.

Closed-Loop Scenarios

The simulated airframe was connected, including the sensor feedback signals. In addition to the open-loop programs, the operational program consisted of the autopilot modes: cruise altitude hold, cruise climb, cruise turn, localizer capture, glideslope capture/track. The land modes consisted of approximately 1200 additional assembly language instructions. In the autopilot modes the aircraft was perturbed from equilibrium flight by initial conditions and control wheel steering commands. The localizer mode was executed every 100 ms, the glideslope modes, every 200 ms and the other modes, every 50 ms.

The open-loop experiments were conducted first. All faults were first injected in the open-loop scenario and only undetected faults were injected in the closed-loop scenarios. This saved time since most faults were detected in the open-loop experiments. Each undetected fault was subsequently injected during an autopilot mode, the purpose being to determine the proportion of faults detected by programs other than the baseline program (it was assumed that the baseline program consisted of the open-loop programs and cruise altitude hold). The same fault was successively injected while executing each of the autopilot modes.

Self-Test

Each FCC contains a self-test program which is normally executed in background. During the open-loop and closed-loop experiments the self-test program was disconnected. In order to determine the coverage of the self-test program and to identify "care faults" all faults which were undetected by both the open-loop and closed-loop programs and a larger sample of detected faults were injected separately, executing only the self-test program.

Test Conditions

- o Faults consisted of permanent S-a-0, S-a-1 and pin inversions, injected on input and output pins of almost every device on the data path and control cards of the CAPS-6 computer (see Appendix A for the device complement). Devices were only excluded in the event that they would not function when the FIIS multiplexers were inserted.
- o No faults were injected in the analog interface hardware.
- o Faults were injected in a single FCC of the dual pair.
- o Only single faults were injected.
- o Faults were detected by comparison monitoring, exclusively. Self-test was not executed in the open-loop and closed-loop scenarios.
- o No faults were injected in the hardware associated with failure detection, i.e., hardware comparators and disengage mechanisms.
- o Detection was recorded at 50 ms intervals (this was the same as the minor frame interval).
- o An undetected fault run was terminated after 15 seconds of real time (i.e., 300 open-loop iterations).
- o Each fault was individually identified.
- o Coverage was tabulated for unweighted and weighted faults. In the latter, faults were weighted in proportion to the failure rate of the device; in the former, all faults were weighted equally. The weights associated with each fault are given in Appendix B.
- o All undetected faults were analyzed and "don't care" faults were eliminated from the tabulations although the number of "don't cares" was tabulated. "Don't care" faults consisted of unused pins or signals that were always high or always low under all operating conditions.

7.2 RESULTS

7.2.1 Open-Loop/Closed-Loop Detection Coverage

These results are given in Tables 1 through 11. Tables 1-4 give the results for unweighted faults; Tables 5-8 for weighted faults; Table 9 summarizes the results for open-loop faults; Table 10 summarizes the results for baseline/auxiliary faults; Table 11 gives a summary of faults activated by autopilot modes. Altogether, 2715 distinct faults were injected in devices of the Data Path and Control Path cards. Of these, 166 were subsequently found to be "don't care" faults (6.11%). In interpreting the tables it is noted that the column "totals" are not the sum of the column entries. The "totals" refer to the column headings, e.g., in Table 1, 2549 = total number of faults injected; 2461 = total number of faults detected.

Table 1: Open-Loop, Unweighted Faults (Invert Faults Included)

Out of 2549 injected "care" faults, 2461 (96.55%) were detected and 88 (3.45%) were undetected. Of the 88 undetected faults, 15 were invert faults. 1.71% of invert faults were undetected whereas 4.37% of stuck-at faults were undetected, indicating that invert faults are less latent than stuck-at faults, as expected. Each of the 88 undetected faults was identified and injected while executing self-test. All of these faults were detected, indicating that they were "care" faults. All micromemory and 2901A faults were detected. This is not surprizing since these devices are highly multiplexed. Detection of input versus output pin faults is approximately equal. This was, at first, surprizing since it was conjectured that input pin faults were more latent than output pin faults. It was realized, subsequently, that most devices of the CAPS-6 computer are sequential, and that output faults of one device are, effectively, input faults of a downstream device. As a consequence, the statistical uncertainty of the results precludes a determination of the relative latency of input and output pins.

Table 2: Closed-Loop, Unweighted Faults (Invert Faults Included)

Each of the 88, undetected, open-loop faults was successively injected while executing each of the six autopilot modes. Thus, the same fault could have been detected while executing several modes, as indeed, the results show. Of the 88 injected faults, 39 (44.32%) were detected, and 49 (55.68%) were undetected. All 88 faults were detected by self-test. It is again noted that 15 of the 88 faults were invert faults.

Table 3: <u>Open-Loop, Unweighted Faults (Invert Faults Excluded)</u>

These results are comparable to those of Table 1 except that invert faults are excluded. Out of 1670 injected "care" faults, 1597 (95.63%) were detected and 7 3(4.37%) were undetected.

Table 4: <u>Closed-Loop, Unweighted Faults (Invert Faults Excluded)</u>

These results are comparable to those of Table 2 except that invert faults are excluded. Out of 73 injected "care" faults, 33 (45.21%) were detected and 40 (54.71%) were undetected.

TABLE 1

Open-Loop
Unweighted Faults (Invert Faults Included)

Device	# Faults Injected	# Faults Detected	# Faults Undetected	# Don't Cares	% Detected	% Undetected
Data Path	1535	1502	33	83	97.85	2.15
Control	1014	959	55	83	94.58	5.42
S-a-0/S-a-1	1670***	1597	73	123	95.63	4.37
Invert	879	864	15	43	98.29	1.71
2901	423	423	0	25	100.	0.0
Micromemory	231	231	0	9	100.	0.0
Input Pins	1805	1741	64	87	96.45	3.55
Output Pins	744	720	24	79	96.77	3.23
Self-Test	1687	1687	0	0	100.	0.0
Totals	2549	2461**	88*	166	96.55**	3.45**

Note: 1) Injected faults do not include "don't cares"
2) % "don't cares" = 6.11
3) * = Detected by self-test
4) ** = Excludes self-test
5) *** = 791 S-a-0, 806 S-a-1

Table 2

Closed-Loop
Unweighted Faults (Invert Faults Included)

Mode	# Faults Injected	# Faults Detected	# Faults Undetected	# Don't Cares	% Detected	% Undetected
Alt Hold	88	13	75	0	14.77	85.23
Climb	88	13	75	0	14.77	85.23
Turn	88	13	75	0	14.77	85.23
Loc/Capt	88	16	72	0	18.18	81.82
GS/Capt	88	11	77	0	12.5	87.5
GS/Track	88	12	76	0	13.64	86.36
Self-Test	88	88	0	0	100.	0.
Totals	88	39*	49*	0	44.32	55.68

Note: 1) * = Detected by self-test

TABLE 3

Open-Loop
Unweighted Faults (Invert Faults Excluded)

Device	# Faults Injected	# Faults Detected	# Faults Undetected	# Don't Cares	% Detected	% Undetected
Data Path	1016	990	26	60	97.44	2.56
Control	654	607	47	63	92.81	7.19
S-a-0/S-a-1	1670	1597	73	123	95.63	4.37
2901	283	283	0	13	100.	0.0
Micromemory	146	146	0	9	100.	0.0
Input Pins	1184	1133	51	69	95.69	4.31
Output Pins	486	464	22	54	95.47	4.53
Self-test	1115	1115	0	0	100.	0.
Totals	1670	1597**	73*	123	95.63	4.37

Notes: 1) % "don't cares" = 6.86
2) * = Detected by self-test
3) ** = Excludes self-test

TABLE 4

Closed-Loop
Unweighted Faults (Invert Faults Excluded)

Mode	# Faults Injected	# Faults Detected	# Faults Undetected	# Don't Cares	% Detected	% Undetected
Alt. Hold	73	11	62	0	15.07	84.93
Climb	73	10	63	0	13.7	86.3
Turn	73	10	63	0	13.7	86.3
Loc/Capt	73	13	60	0	17.81	82.12
GS/Capt	73	9	64	0	12.33	87.67
GS/Track	73	12	61	0	16.44	83.56
Self-Test	73	. 73	0	0	100.	0.0
Totals	73	33*	40*	0	45.21	54.79

Notes: 1) * = Detected by self-test

Table 5: Open-Loop, Weighted Faults (Invert Faults Included)

These results are comparable to those of Table 1 except that faults are weighted. Out of 282,345 injected "care" faults, 274,926 (97.37%) were detected and 741 9(2.62%) were undetected.

Table 6: Closed-Loop, Weighted Faults (Invert Faults Included)

These results are comparable to those of Table 2 except that faults are weighted. Out of 7419 injected "care" faults, 3591 (48.4%) were detected and 3828 (51.6%) were undetected.

Table 7: Open-Loop, Weighted Faults (Invert Faults Excluded)

These results are comparable to those of Table 3 except that faults are weighted. Out of 185,253 injected "care" faults, 178,787 (96.51%) were detected and 6466 (3.49%) were undetected.

Table 8: Closed-Loop, Weighted Faults (Invert Faults Excluded)

These results are comparable to those of Table 4 except that faults are weighted. Out of 6466 injected "care" faults, 3063 (47.37%) were detected and 3403 (52.63%) were undetected.

Table 9: Summary of Open-Loop Faults

This table shows the relative differences between detection coverage of 1) unweighted vs weighted faults and 2) invert faults vs non-invert faults. In both cases, the differences are of the order of 1%.

TABLE 5

Open-Loop
Weighted Faults (Invert Faults Included)

Device	# Faults Injected	# Faults Detected	# Faults Undetected	# Don't Cares	% Detected	% Undetected
S-a-0/S-a-1	185,253	178,787	6466	15,444	96.51	3.49
Invert	97,092	96,139	953	5,964	99.02	0.98
2901	60,912	60,912	0	3,600	100.	0.0
Micromemory	24,255	24,255	0	945	100.	0.0
Input Pins	192,350	187,390	4960	9,913	97.42	2.58
Output Pins	89,995	87,536	2459	11,495	97.27	2.73
Self-Test	220,482	220,482	0	0	100.	0.0
Totals	282,345	274,926**	7419*	21,408	97.37	2.63

Notes: 1) % "don't cares" = 7.05
2) * = Detected by self-test
3) ** = Excludes self-test

TABLE 6

Closed-Loop
Weighted Faults (Invert Faults Included)

Mode	# Faults Injected	# Faults Detected	# Faults Undetected	# Don't Cares	% Detected	% Undetected
Alt. Hold	7419	1278	6141	0	17.23	82.77
Climb	7419	1135	6284	0	15.3	84.7
Turn	7419	1100	6319	0	14.83	85.17
Loc/Capt	7419	1376	6043	0	18.55	81.45
GS/Capt	7419	793	6626	0	10.69	89.31
GS/Track	7419	1188	6231	0	16.0	84.0
Self-Test	7419	7419	0	0	100.	0.0
Totals	7419	3591*	3828*	0	48.4	51.6

Notes: 1) * = Detected by self-test

TABLE 7

Open-Loop
Weighted Faults (Invert Faults Excluded)

Device	# Faults Injected	# Faults Detected	# Faults Undetected	# Don't Cares	% Detected	% Undetected
S-a-0/S-a-1	185,253	178,787	6466	15,444	96.51	3.49
2901	40,752	40,752	0	1,872	100.	0.0
Micromemory	15,330	15,330	0	945	100.	0.0
Input Pins	126,179	122,105	4074	7,476	96.77	3.23
Output Pins	59,074	56,682	2394	7,968	95.95	4.05
Self-Test	146,375	146,375	0	0	100.	0.0
Totals	185,253	178,787**	6466*	15,444	96.51	3.49

Notes: 1) % "don't care" faults = 7.7
2) * = Detected by self-test
3) ** = Excludes self-test

TABLE 8

Closed-Loop
Weighted Faults (Invert Faults Excluded)

Mode	# Faults Injected	# Faults Detected	# Faults Undetected	# Don't Cares	% Detected	% Undetected
Alt. Hold	6466	1087	5379	0	16.81	83.19
Climb	6466	871	5595	0	13.47	86.53
Turn	6466	881	5585	0	13.63	86.37
Loc/Capt	6466	1112	5354	0	17.2	82.8
GS/Capt	6466	647	5819	0	10.0	90.0
GS/Track	6466	1188	5278	0	18.37	81.63
Self-Test	6466	6466	0	0	100.	0.0
Totals	6466	3063*	3403*	0	47.37	52.63

Notes: 1) * = Detected by self-test

TABLE 9

Summary of Open-Loop Faults

	Invert Faults Included Unweighted	Invert Faults Included Weighted	Invert Faults Excluded Unweighted	Invert Faults Excluded Weighted
% Detected	96.55	97.37	95.63	96.51
% Undetected	3.45	2.63	4.37	3.49

7.2.2 Detection Coverage of Baseline and Auxiliary Programs

The reader will recall (Section 4.1) that a baseline program is a set of software that is executed continuously. In obtaining the following estimates, it was assumed that the baseline program consisted of the open-loop programs together with altitude hold. Thus, the auxiliary progrms consisted of the remaining autopilot modes.

1a. Unweighted Faults, Invert Faults Included

Total Faults Injected = 2549
Detected by Baseline =2474;
Undetected = 75;
%Detected = 97.07
Undetected by Baseline but detected by Auxiliary = 26 (1.02%)
Undetected by Baseline and undetected by Auxiliary = 49 (1.92%)

1b. Unweighted Faults, Invert Faults Excluded

Total Faults Injected = 1670
Detected by Baseline = 1608;
Undetected = 62;
%Detected = 96.287
Undetected by Baseline but detected by Auxiliary = 22 (1.317%)
Undetected by Baseline and undetected by Auxiliary = 40 (2.395%)

2a. Weighted Faults, Invert Faults Included

Total Faults Injected = 282,345
Detected by Baseline = 276,204;
Undetected = 6141;
%Detected = 97.83
Undetected by Baseline but detected by Auxiliary = 2313 (0.819%)
Undetected by Baseline and undetected by Auxiliary = 3828 (1.92%)

2b. Weighted Faults, Invert Faults Excluded

Total Faults Injected = 185,253
Detected by Baseline = 179,874;
Undetected = 5379;
%Detected = 97.1
Undetected by Baseline but detected by Auxiliary = 1976 (1.65%)
Undetected by Baseline and undetected by Auxiliary = 3403 (1.84%)

Table 10: Summary of Baseline/Auxiliary Faults

These results indicate that most faults are detected while executing the baseline program. However, there is a significant proportion of faults not detected by the baseline and auxiliary programs, e.g., 2%, approximately. The effect of these undetected faults remains to be determined. It is noted that self-test detected all of these faults.

TABLE 10

Summary of Baseline/Auxiliary Faults

	Invert Faults Included		Invert Faults Excluded	
	Unweighted	Weighted	Unweighted	Weighted
Detected by Baseline	2474 (97.07%)	276,204 (97.83%)	1608 (96.25%)	179,874 (97.1%)
Not Detected by Baseline but Detected by Auxiliary	26 (1.02%)	2,313 (0.819%)	22 (1.317%)	1,976 (1.65%)
Not Detected by Baseline and not Detected by Auxiliary	49 (1.92%)	3,828 (1.92%)	40 (2.395%)	3,403 (1.84%)

7.2.3 Latency Histograms

The fault injection data was organized to give probability density functions (pdf) of detection (%) versus time for a variety of different fault sets. Histograms of the pdf's are given in Figures 4 to 21. The figures are organized as follows:

Figures 4 to 12:	Open-Loop, Unweighted Faults (Invert Faults Excluded)
Figure 4:	All Faults
Figure 5:	Data Path
Figure 6:	Control Card
Figure 7:	2901A
Figure 8:	Micromemory
Figure 9:	S-a-0
Figure 10:	S-a-1
Figure 11:	Input Pins
Figure 12:	Output Pins
Figure 13:	Open-Loop, Unweighted Invert Faults
Figure 14:	All Open-Loop, Unweighted Faults (Invert Faults Included)
Figure 15:	All Open-Loop, Weighted Faults (Invert Faults Excluded)
Figures 16 to 21:	Closed-Loop, Unweighted Faults (Invert Faults Excluded)
Figure 16:	Alt. Hold
Figure 17:	Climb
Figure 18:	Turn
Figure 19:	Loc/Capt
Figure 20:	GS/Capt
Figure 21:	GS/Trk

Although a few faults (0.3%) were occasionally detected between 1 and 6.5 seconds, the time scales were terminated at 5000 ms. The percentage of faults that were not detected is also indicated on each histogram. All of the histograms show that most detected faults are detected within 500ms of their occurrence, irrespective of the types of faults injected, e.g., S-a-0, S-a-1, invert. A comparison of Figures 9, 10, 13 indicates that the histograms for S-a-0, S-a-1 and Invert faults are remarkably similar. The histogram for all faults (Figure 4) reflects this similarity. The alternating peaks and valleys, which are characteristic of all of the histograms, reflect the complexity of error propagation. From Figures 16 through 21 it can be seen that closed-loop faults require somewhat longer times to produce errors.

Figure 4. All Open-Loop, Unweighted Faults (Invert Faults Excluded)

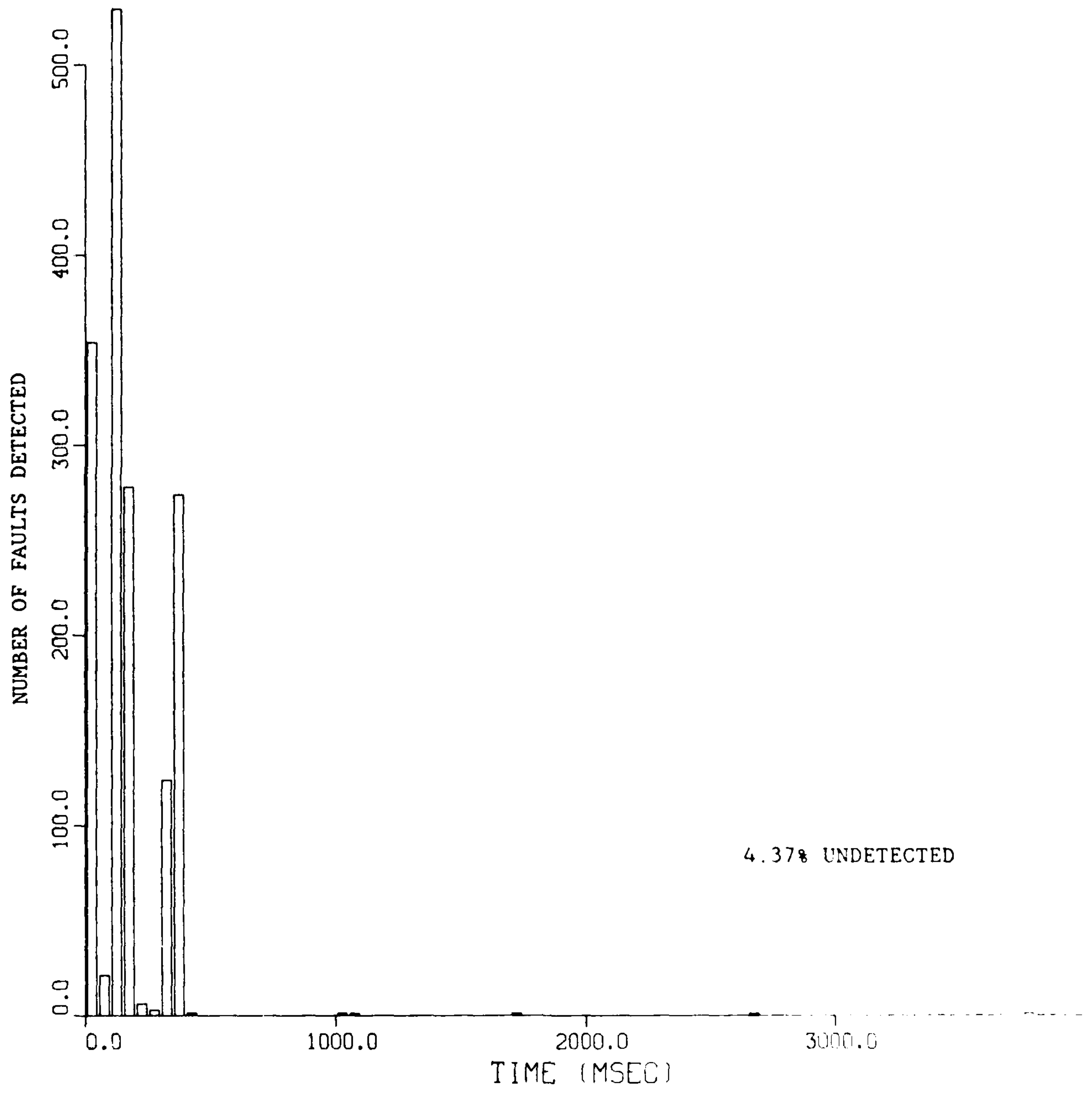


Figure 5. Open-Loop, Unweighted Data Path Faults (Invert Faults Excluded)

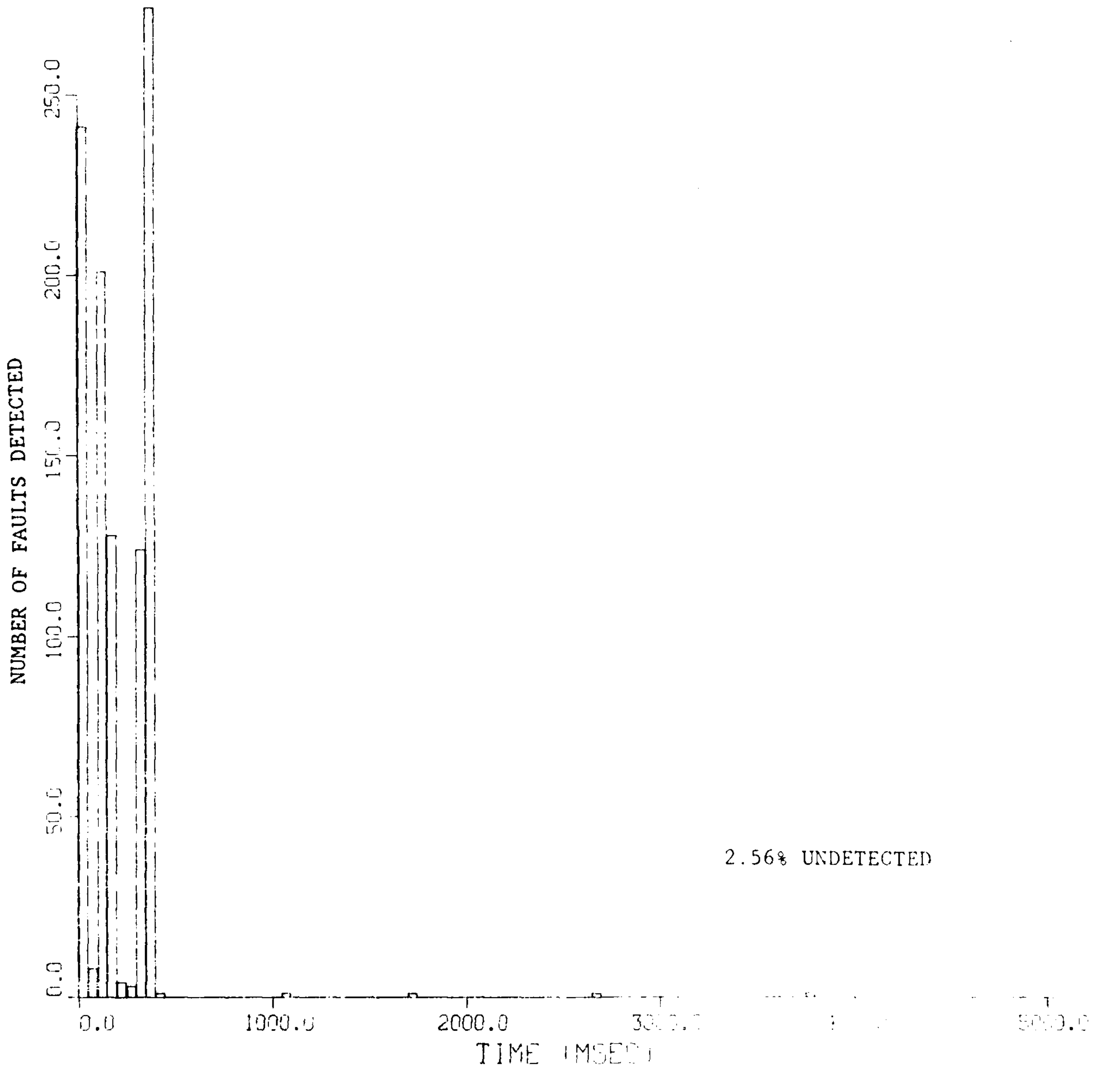


Figure 6. Open-Loop, Unweighted Control Card Faults
(Invert Faults Excluded)

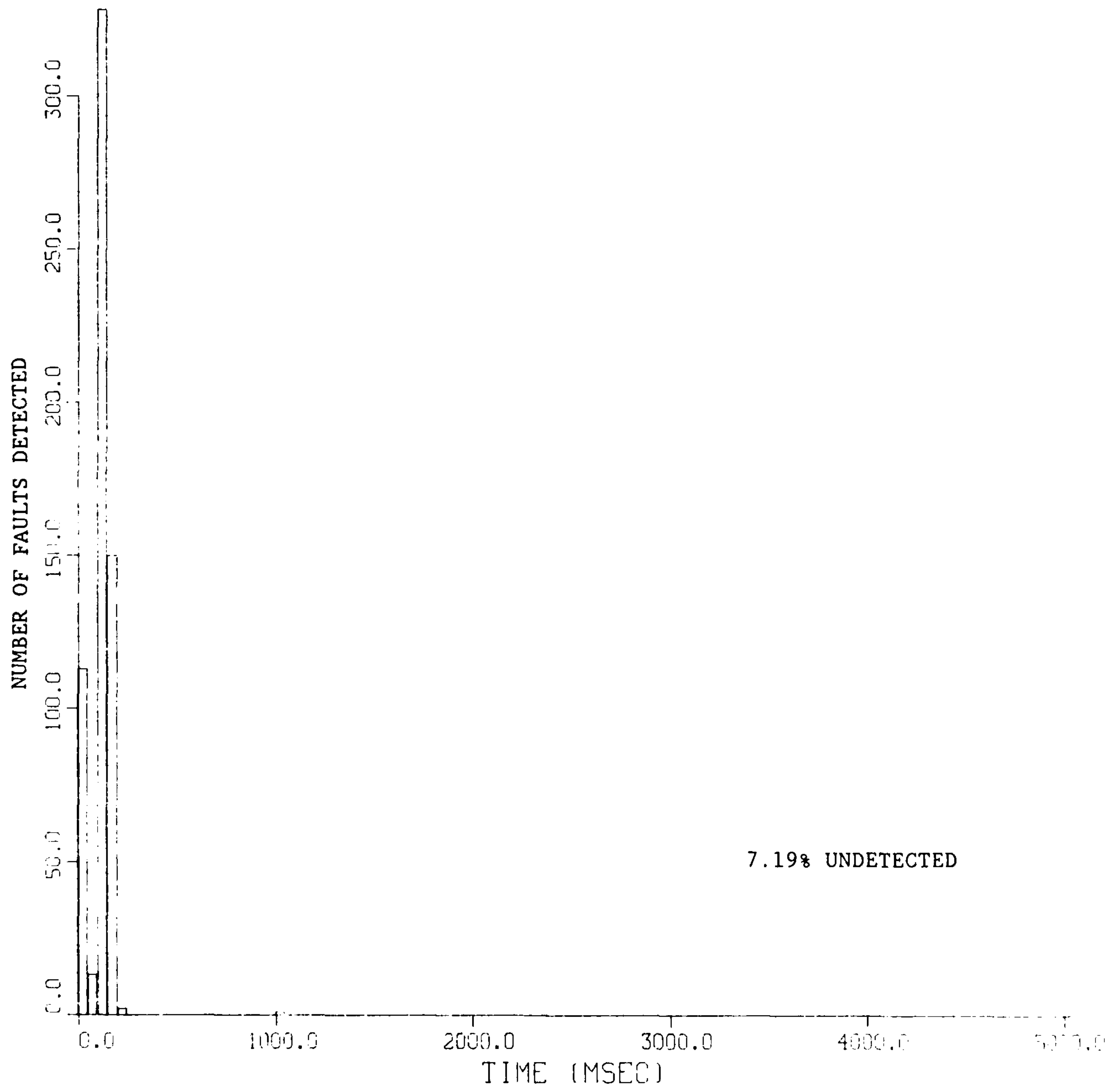


Figure 7. Open-Loop, Unweighted 2901A Faults (Invert Faults Excluded)

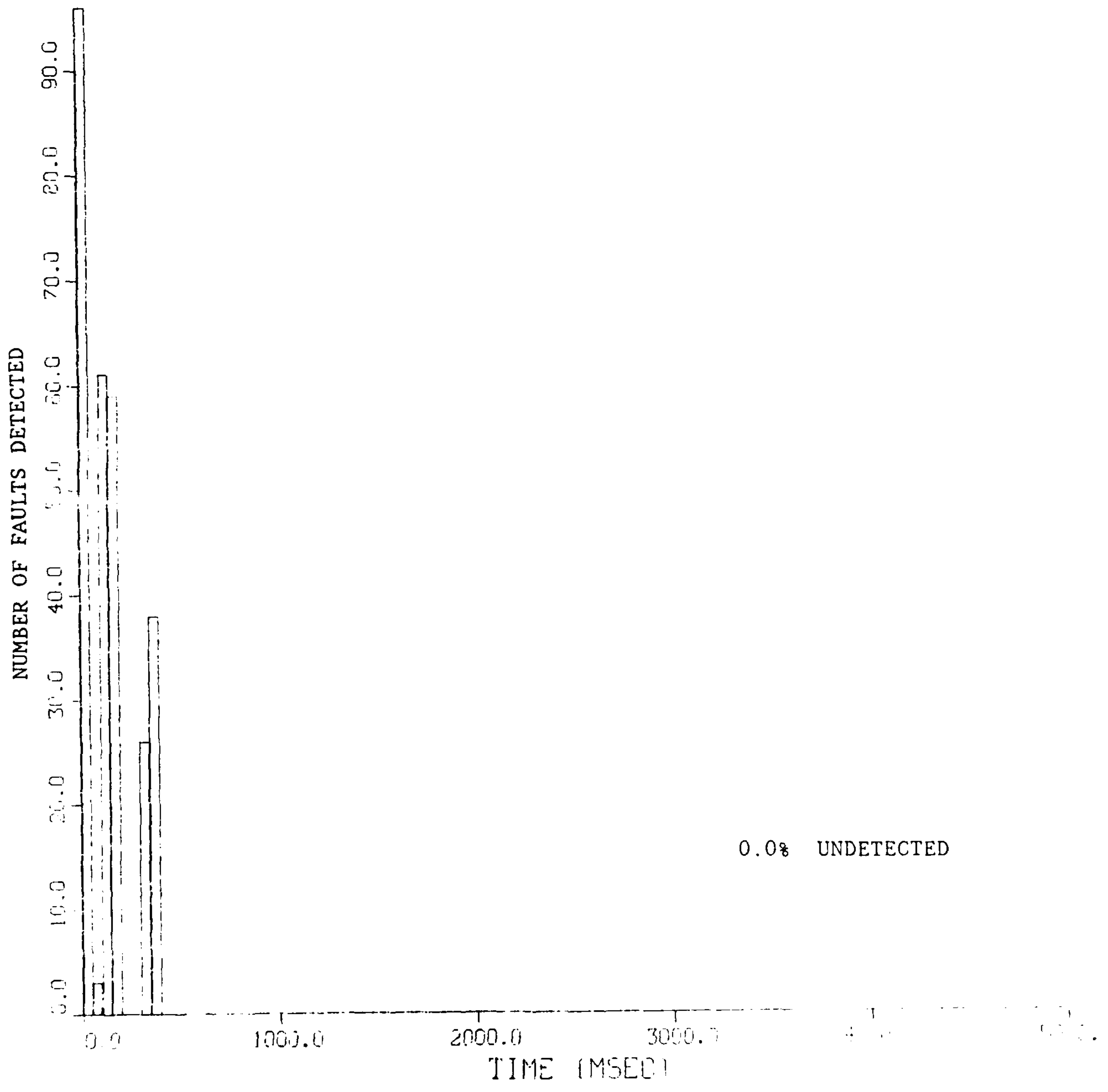


Figure 8. Open-Loop, Unweighted Micromemory Faults
(Invert Faults Excluded)

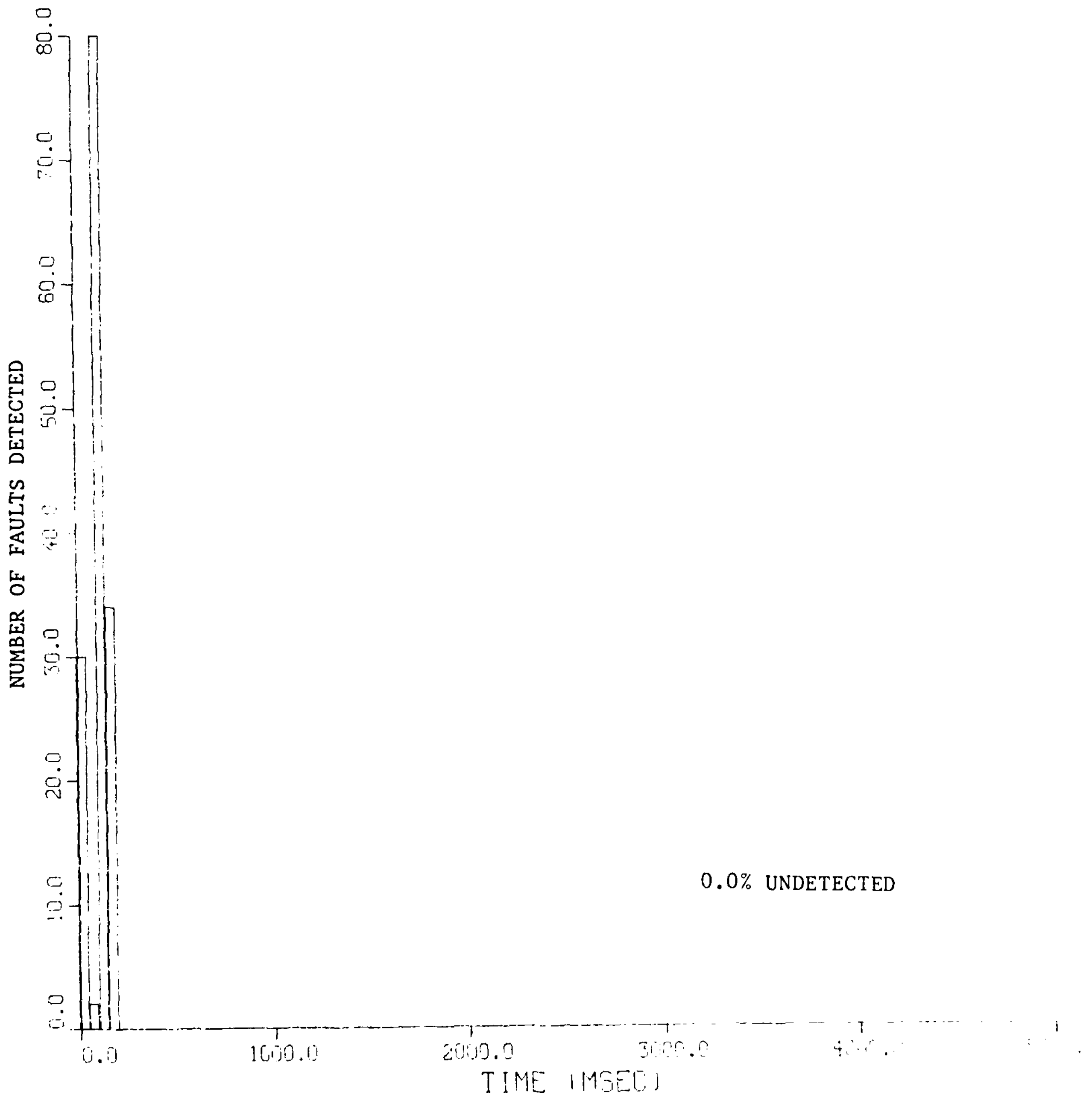


Figure 9. Open-Loop, Unweighted S-a-0 Faults (Invert Faults Excluded)

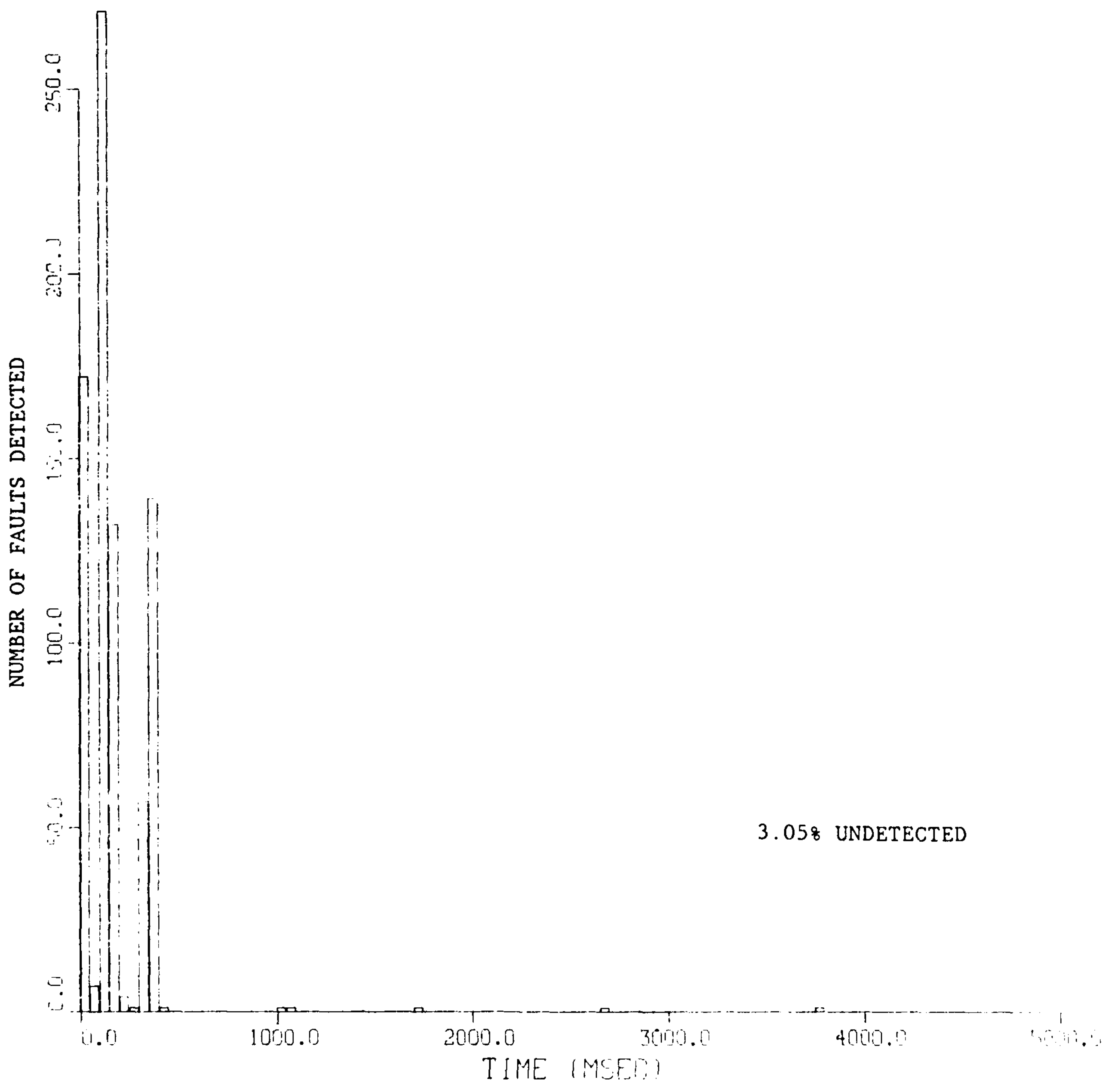


Figure 10. Open-Loop, Unweighted S-a-1 Faults (Invert Faults Excluded)

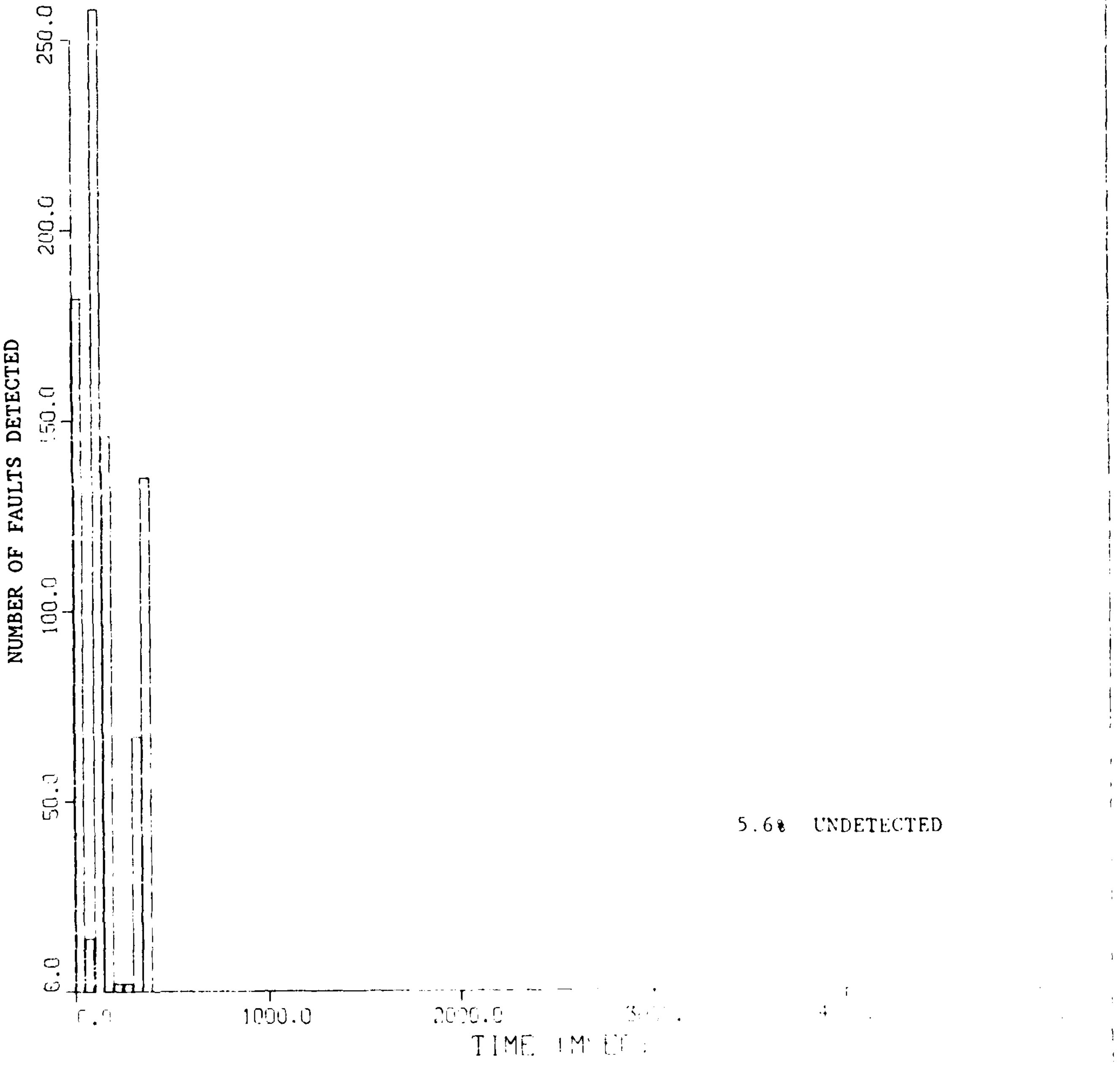


Figure 11. Open-Loop, Unweighted Input Pin Faults
(Invert Faults Excluded)

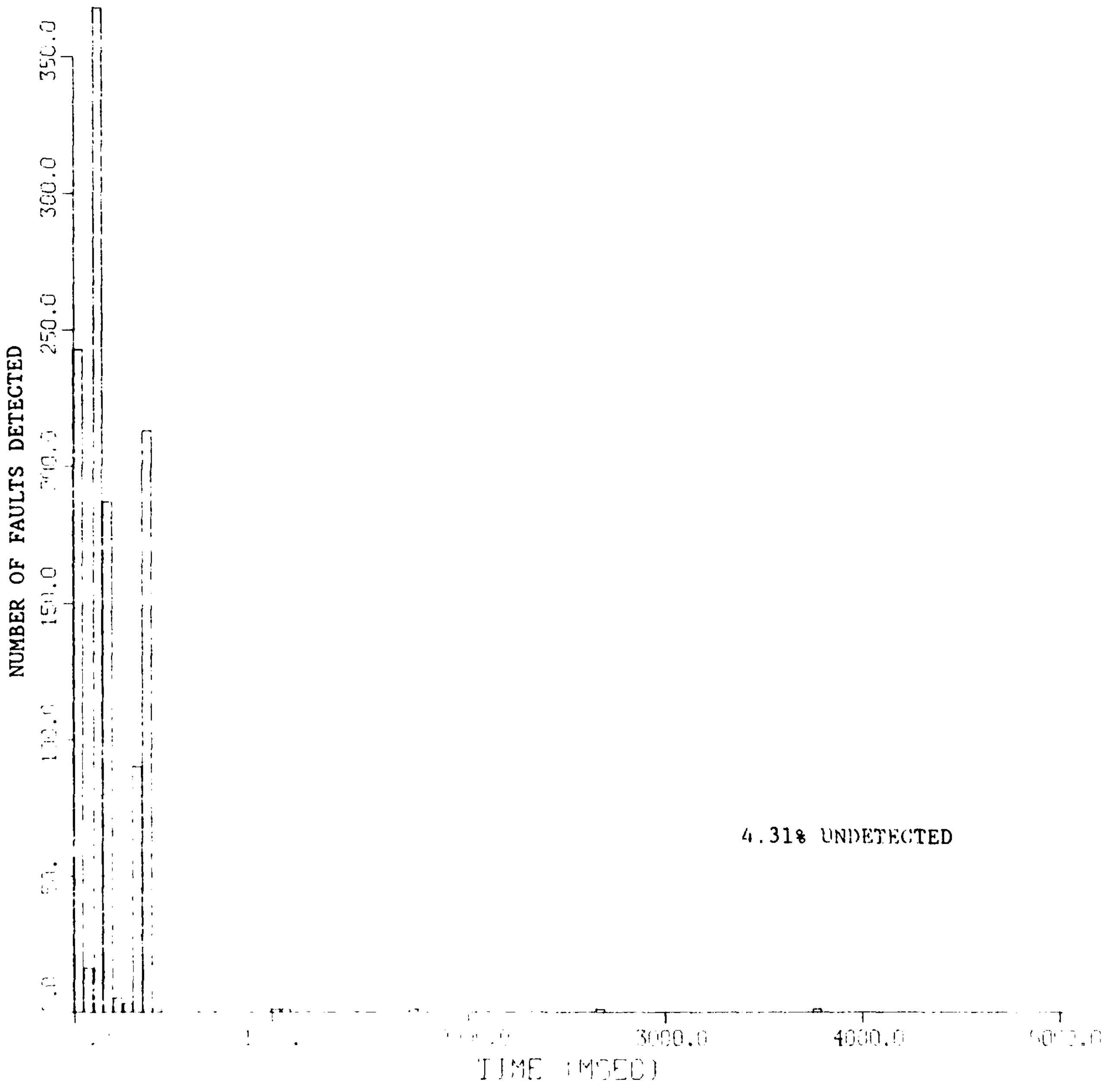


Figure 12. Open-Loop, Unweighted Output Pin Faults (Invert Faults Excluded)

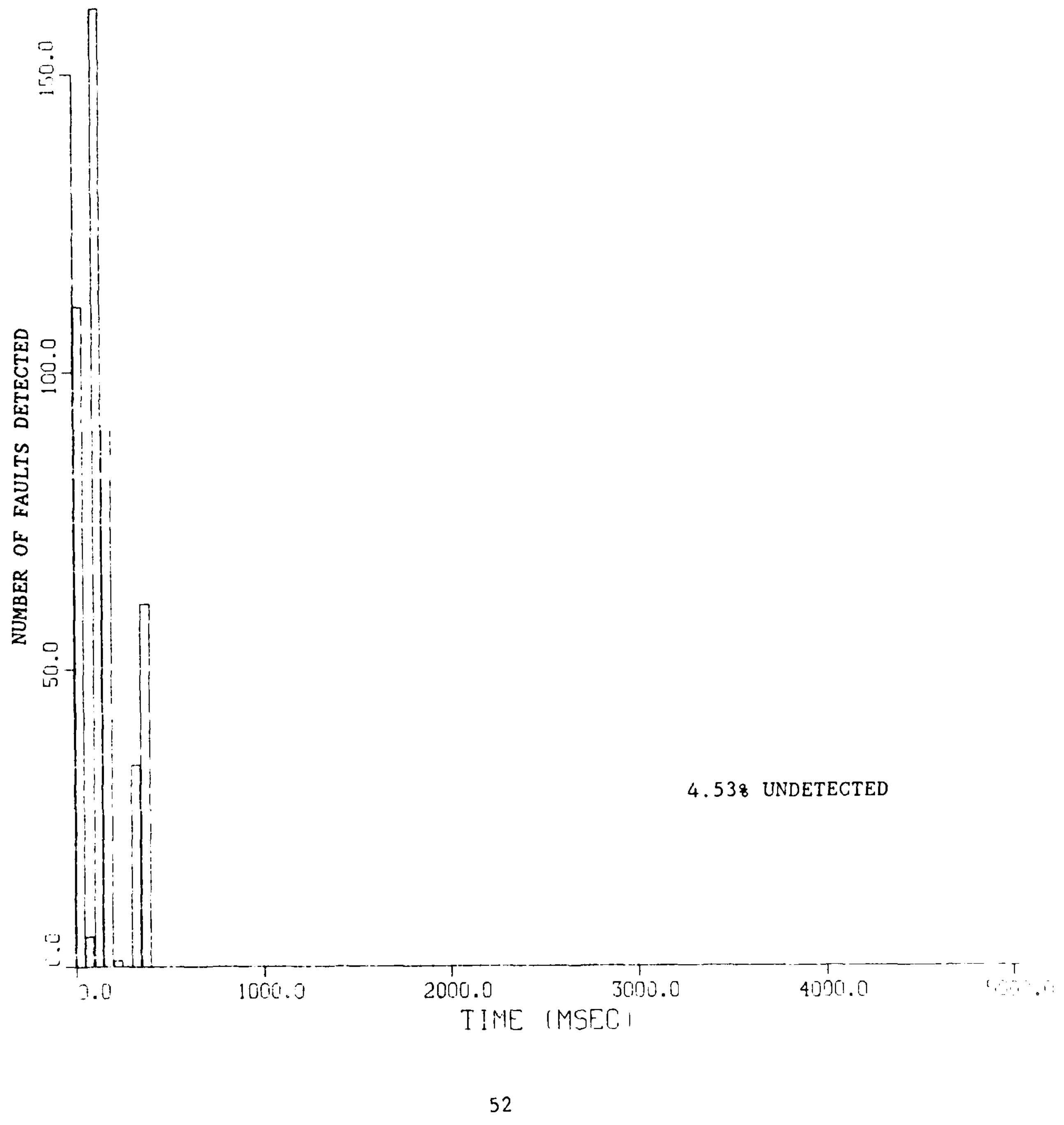


Figure 13. Open-Loop, Unweighted Invert Faults

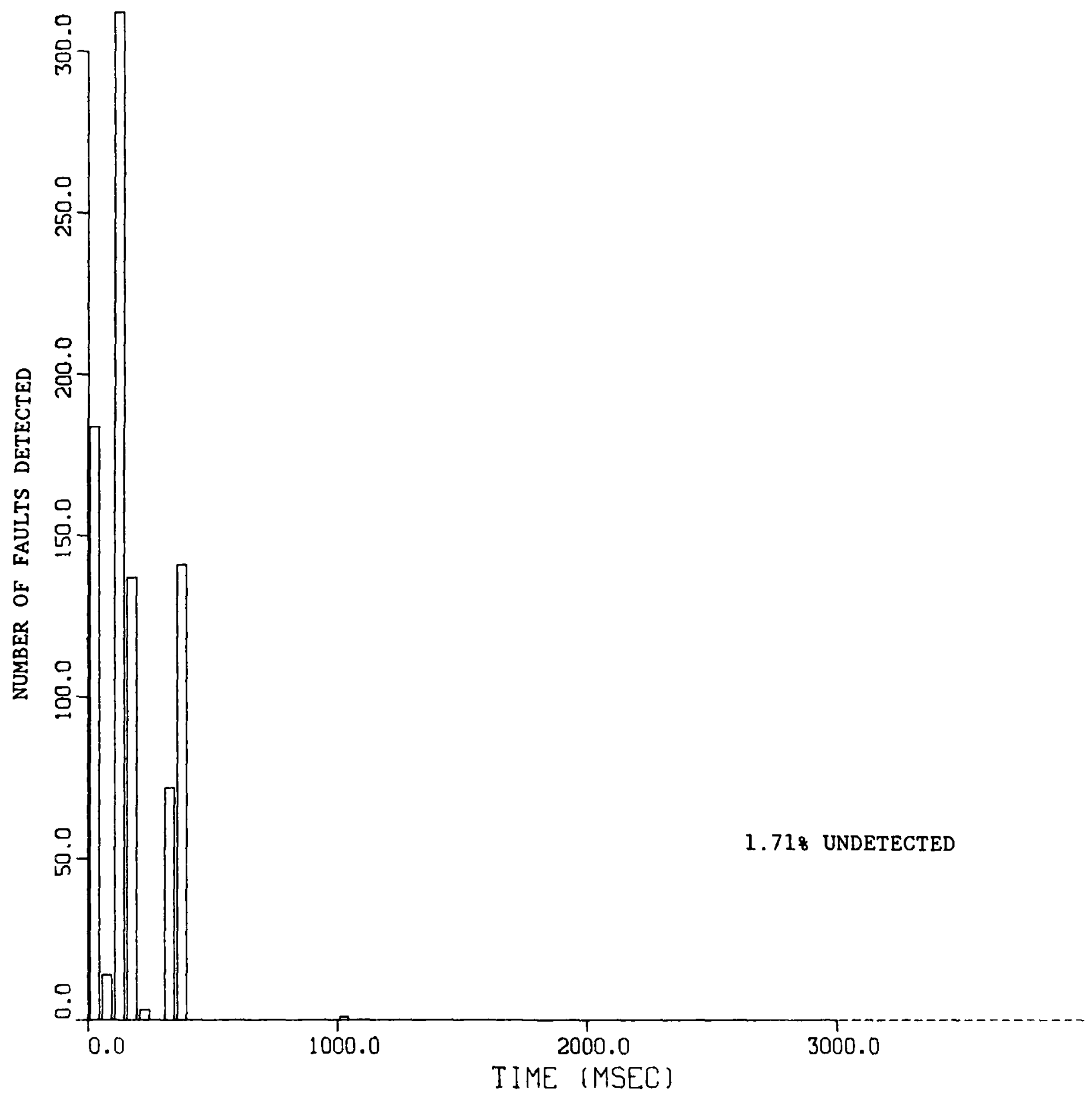


Figure 14. All Open-Loop, Unweighted Faults (Invert Faults Included)

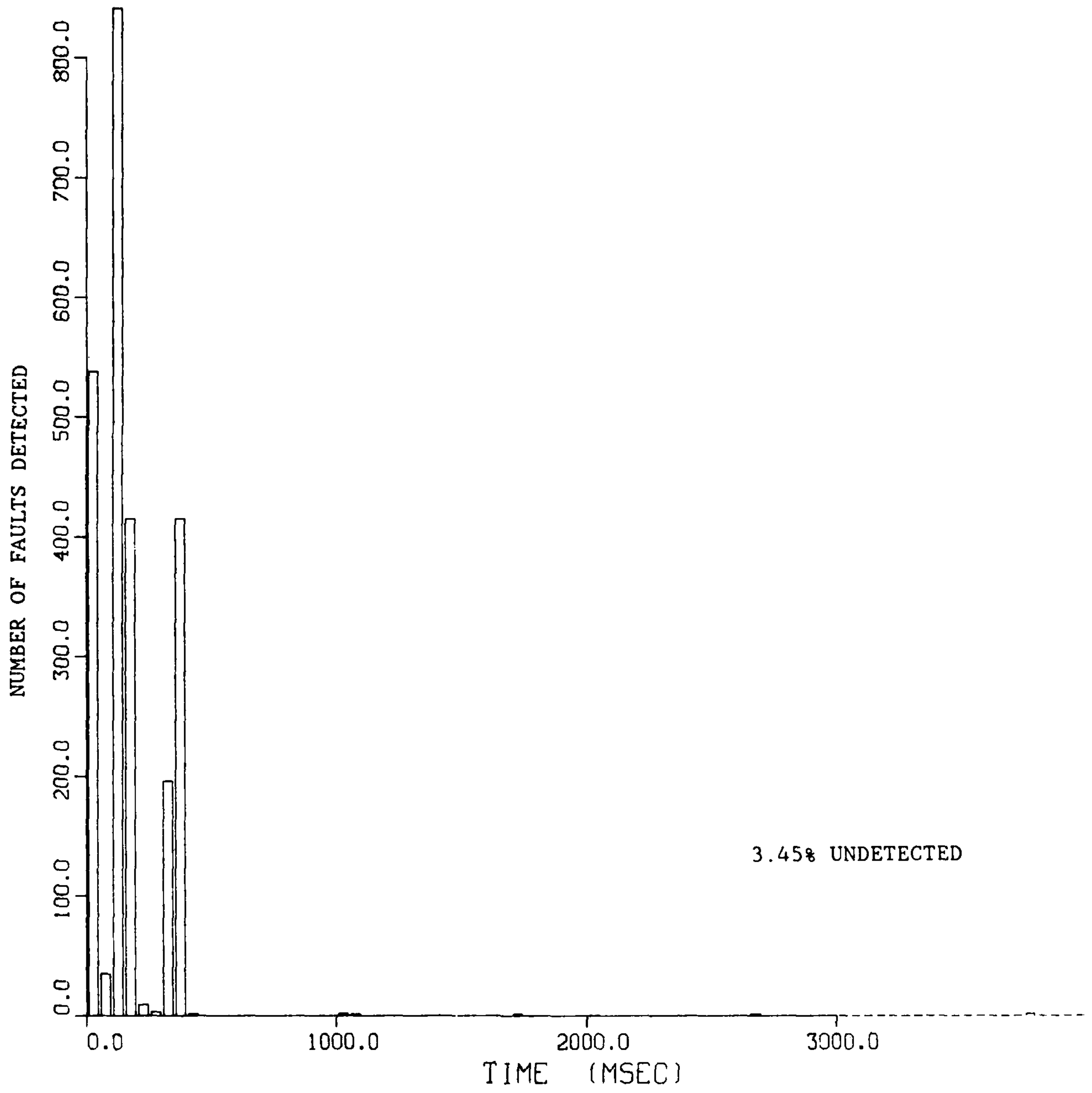


Figure 15. All Open-Loop, Weighted Faults (Invert Faults Excluded)

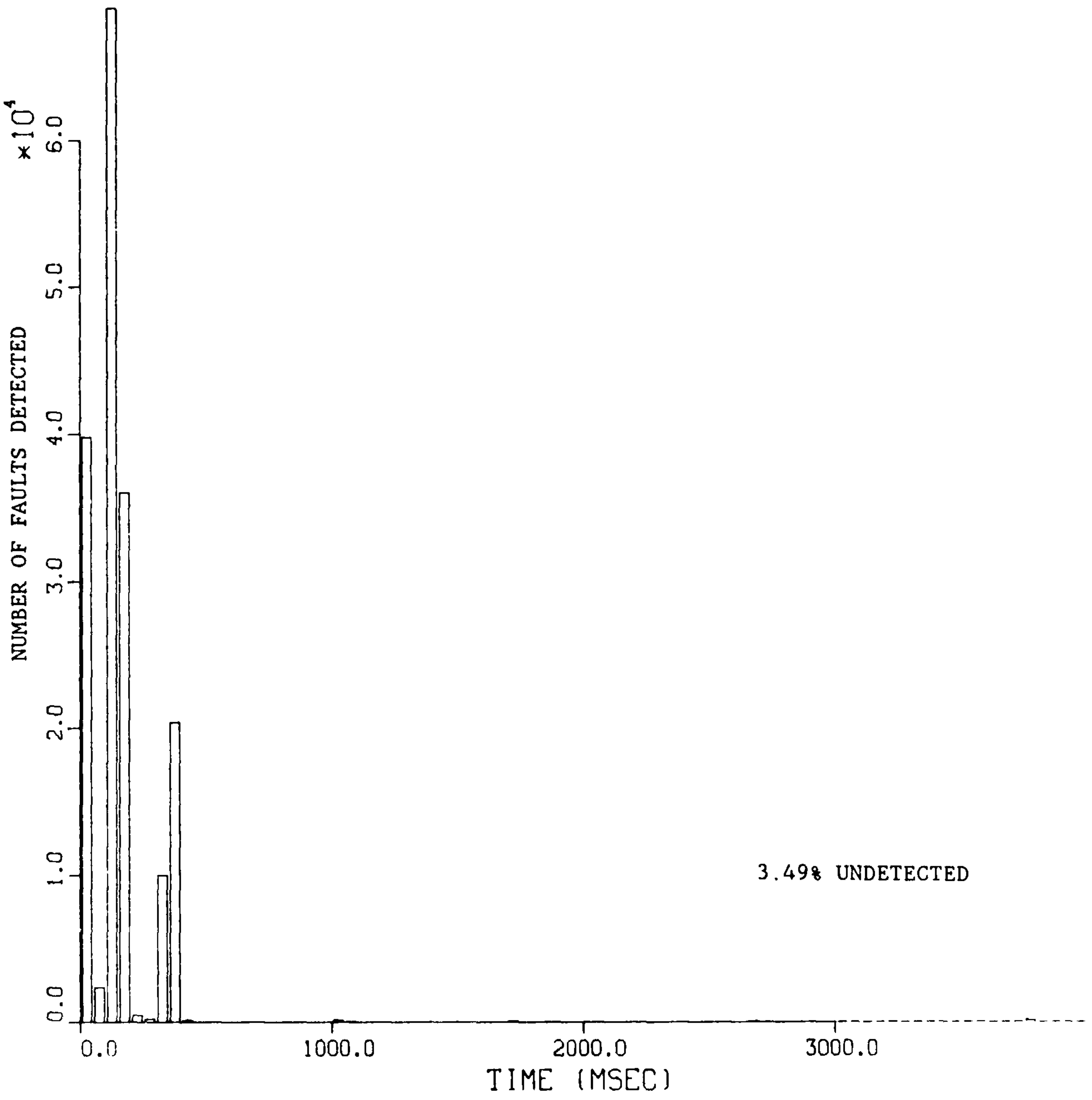


Figure 16. Closed-Loop, Unweighted Altitude Hold Faults
(Invert Faults Excluded)

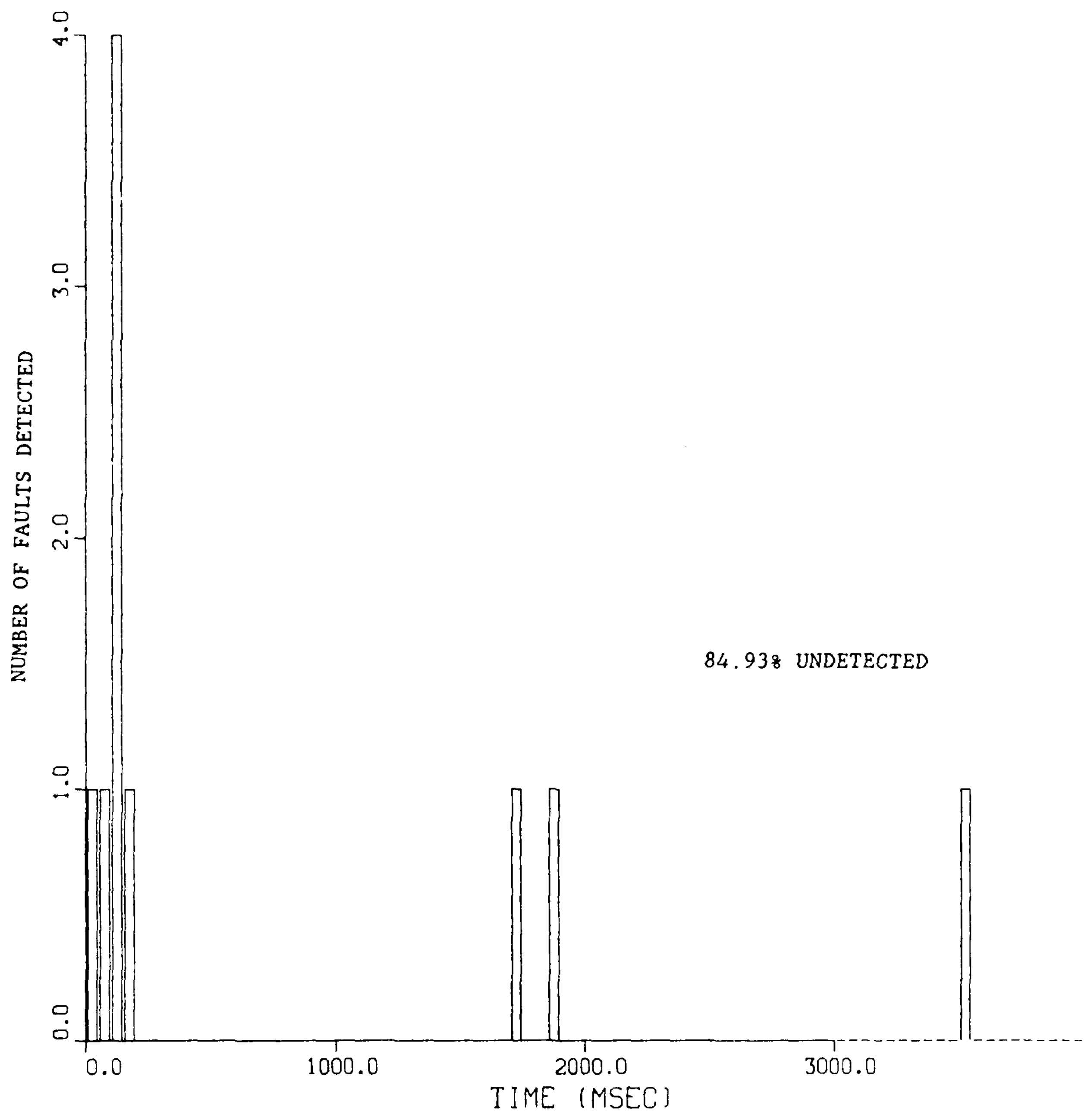


Figure 17. Closed-Loop, Unweighted Climb Faults
(Invert Faults Excluded)

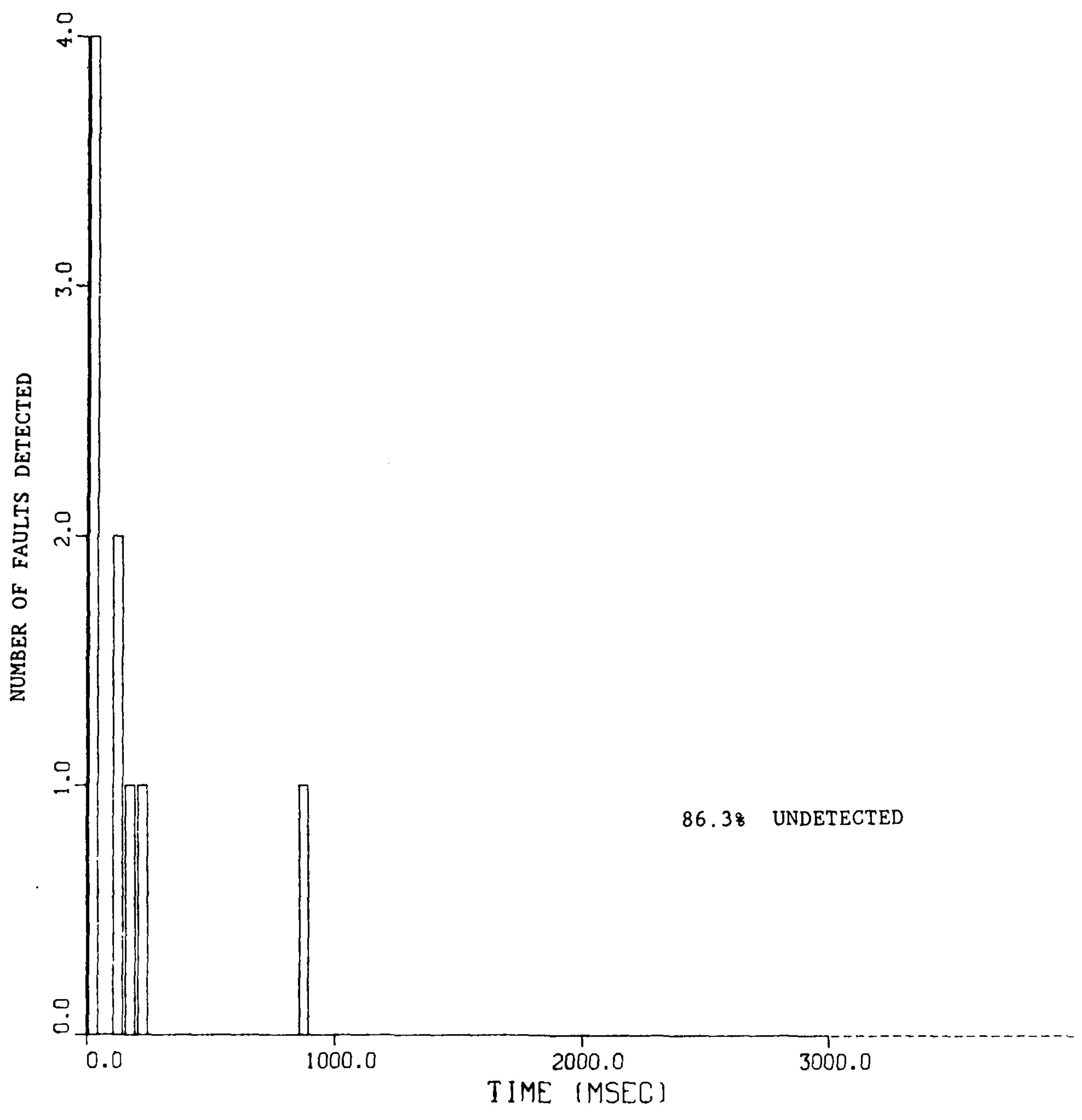


Figure 18. Closed-Loop, Unweighted Turn Faults
(Invert Faults Excluded)

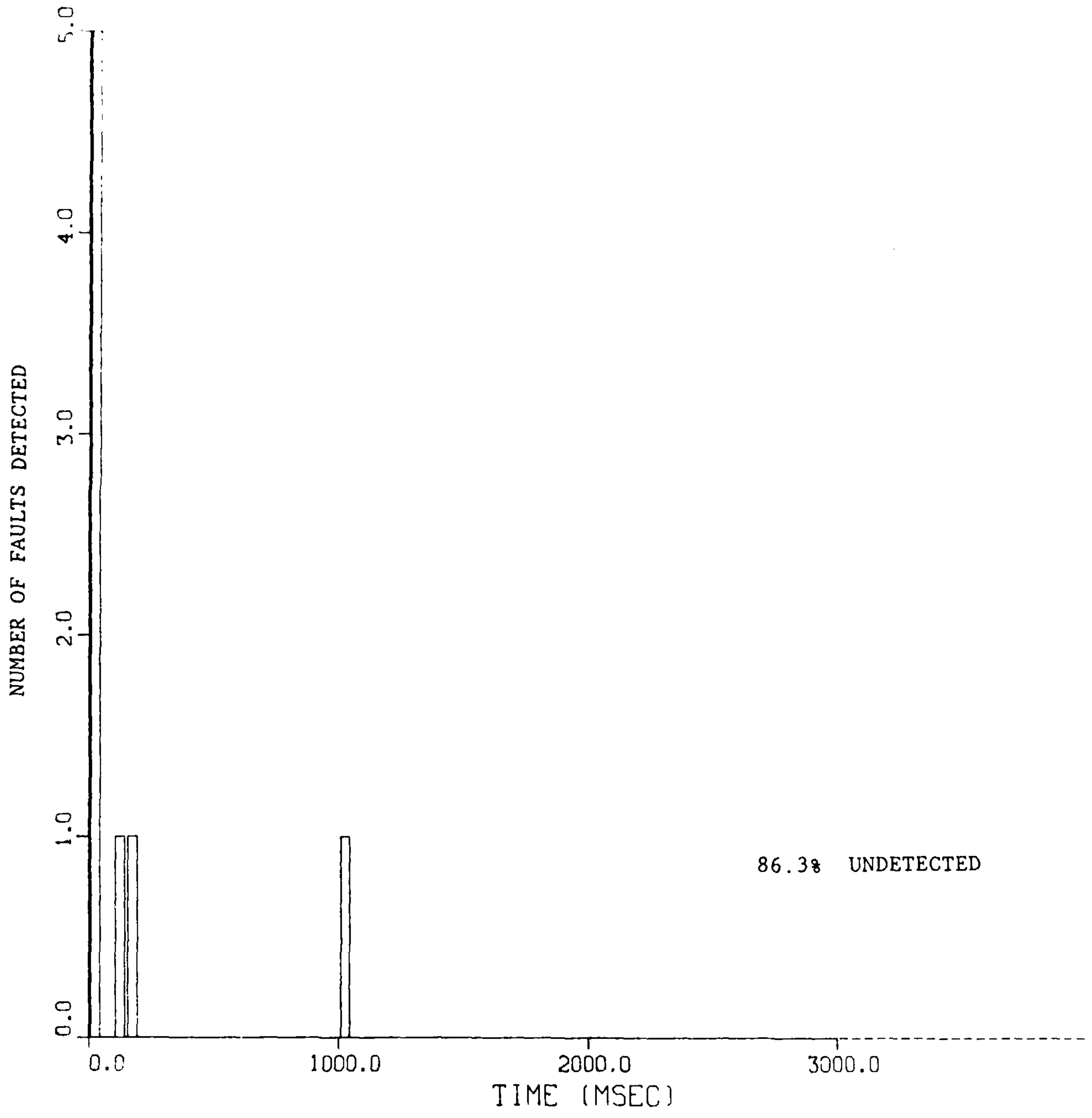


Figure 19. Closed-Loop, Unweighted Localizer Capture Faults (Invert Faults Excluded)

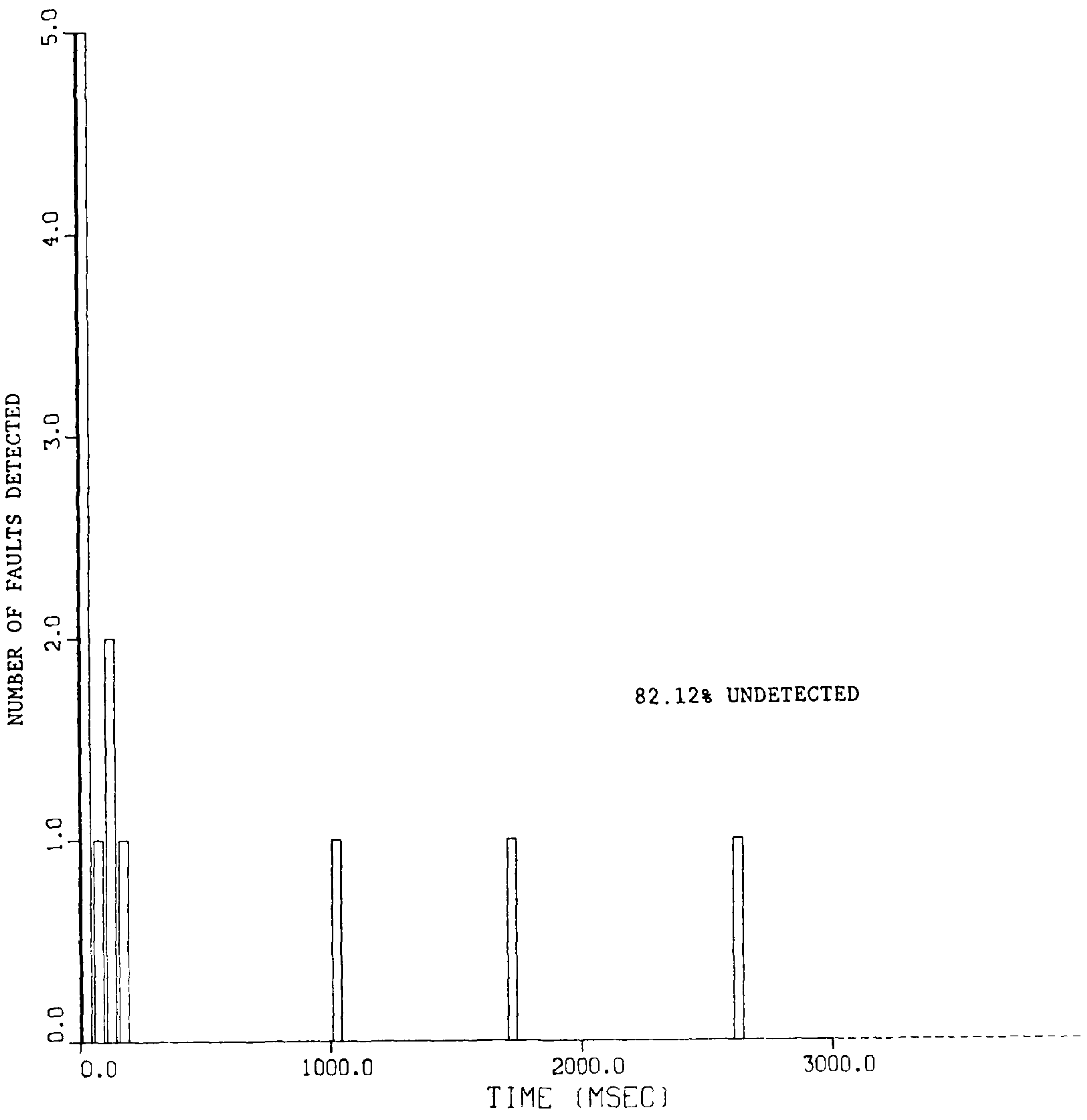


Figure 20. Closed-Loop, Unweighted Glideslope Capture Faults
(Invert Faults Excluded)

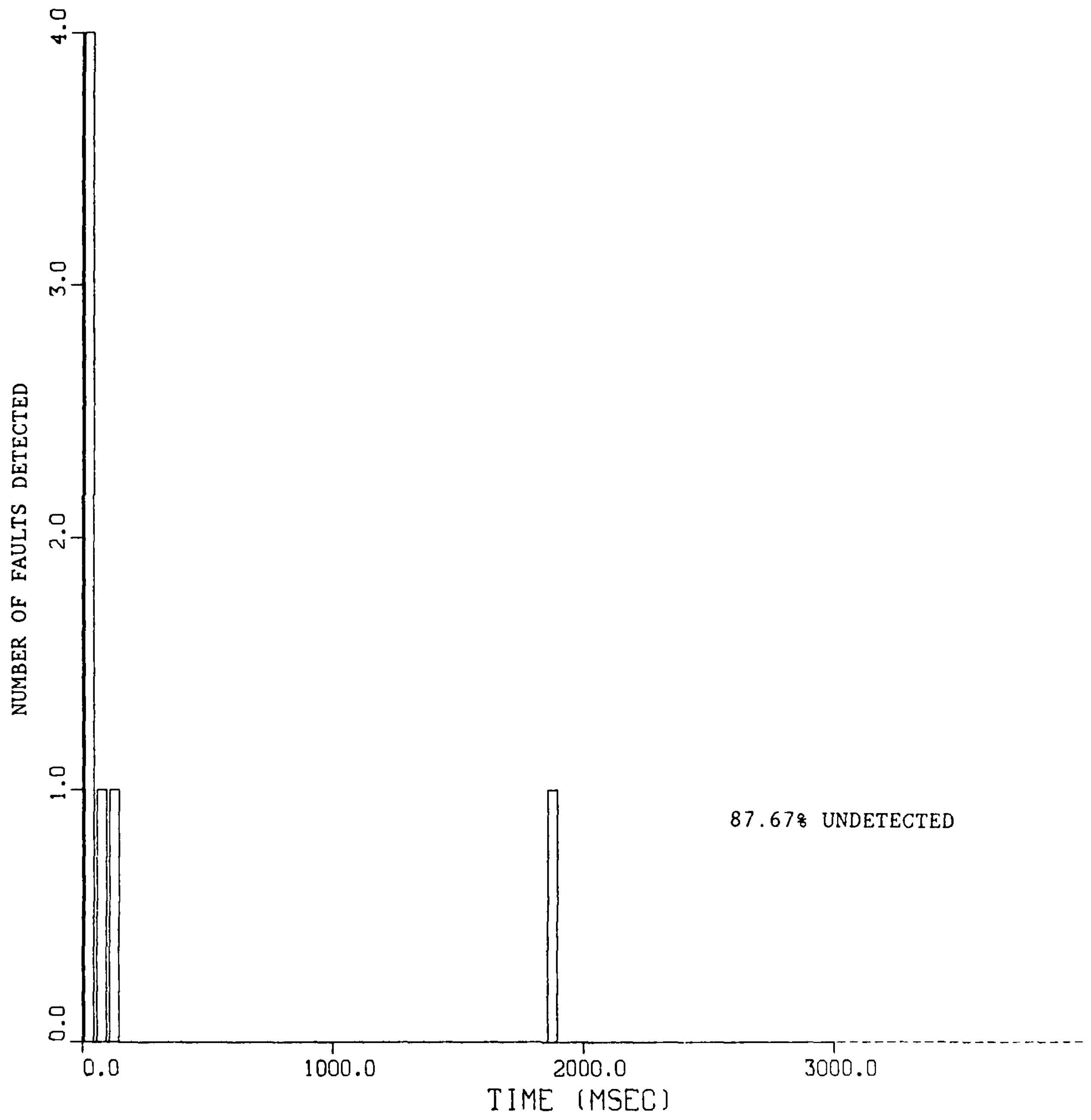
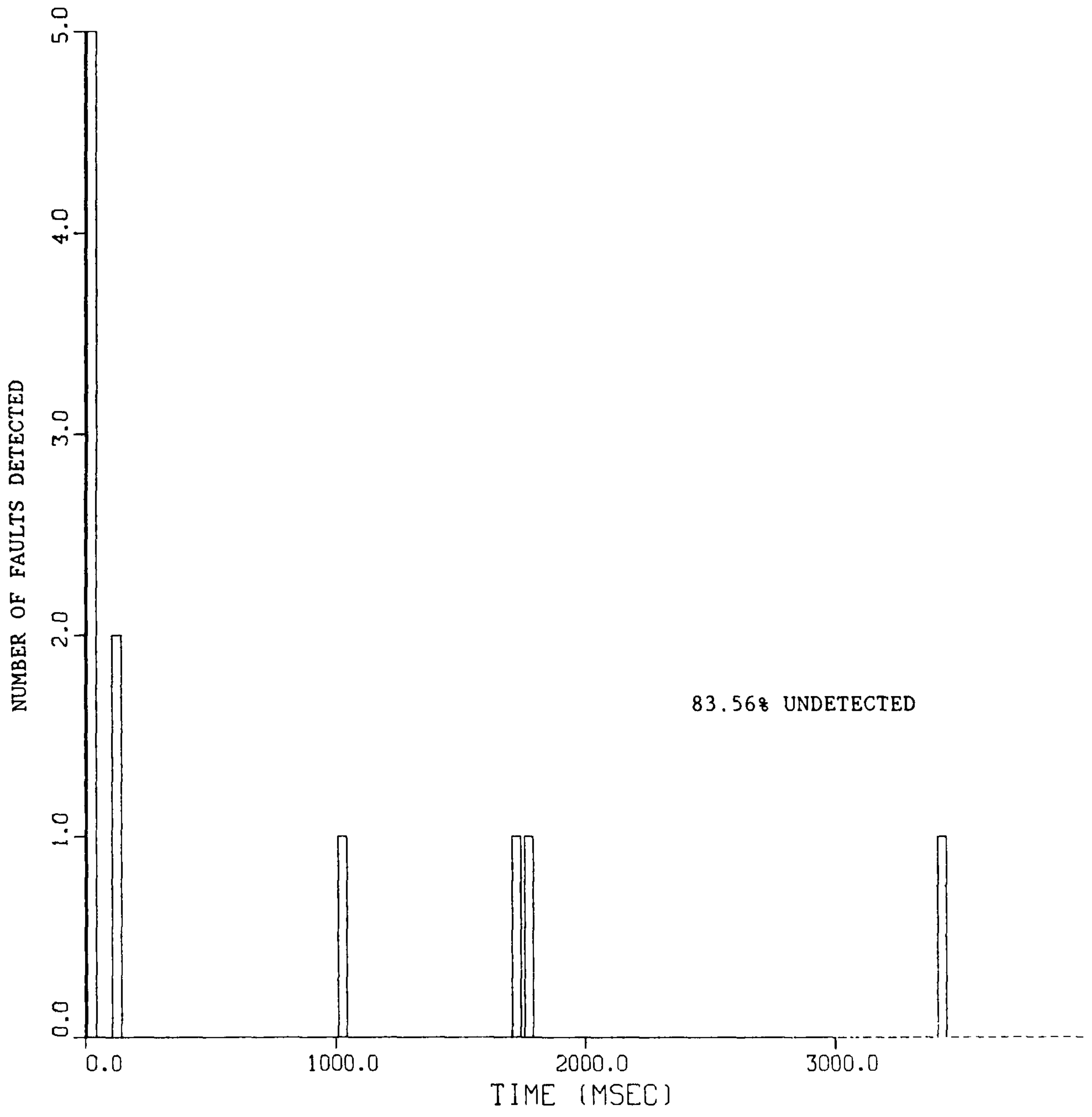


Figure 21. Closed-Loop, Unweighted Glideslope Track Faults (Invert Faults Excluded)



THIS PAGE LEFT INTENTIONALLY BLANK.

8.0 COMPARISON WITH PREVIOUS FAULT INJECTION EXPERIMENTS

Test Ground Rules/Procedures/Test Conditions

1) Target Hardware

Bendix:	Bendix BDX-930 digital processor.
CSDL:	FTMP, which included the CAPS-6 digital processor.
FIIS:	CAPS-6 digital processor.

2) Simulation Technique

Bendix:	The processor was simulated in software at the gate-level and faults were injected at gate nodes.
CSDL:	Faults were injected into the real hardware, at the pin-level.
FIIS:	Faults were injected into the real hardware, at the pin-level.

3) Sensor Inputs

Bendix:	Sensor inputs were selected at random at the start of each frame.
CSDL:	Sensor inputs were normal sensor values, obtained from a simulated airframe.
FIIS:	Sensor inputs were constant in the open-loop scenario and normal sensor values, obtained from a simulated airframe, in the closed-loop scenarios.

4) Fault Detection

Bendix:	Faults were defined as "detected" if they produced an error in a computed output.
CSDL:	Faults were detected by the Bus Guardian Units (comparators) and by background self-test programs.
FIIS:	During the FCS experiments, faults were detected by software and hardware comparators. Background self-test was disconnected.

5) Types of Faults

Bendix:	Permanent S-a-0/S-a-1. Weighted faults, only. Faults injected, one at a time.
CSDL:	Permanent S-a-0/S-a-1. Unweighted faults, only. Faults injected, one at a time.
FIIS:	Permanent S-a-0/S-a-1 and invert. Weighted and unweighted faults. Faults injected, one at a time.

6) I/O Hardware

Bendix: Analog I/O hardware was not simulated.
CSDL: Faults were not injected in the analog I/O hardware.
FIIS: Faults were not injected in the analog I/O hardware.

7) Failure Detection Hardware

Bendix: Not simulated; no faults injected.
CSDL: Faults were injected in the BGUs.
FIIS: No faults injected.

8) Termination of an Undetected Fault Run

Bendix: After eight repetitions.
CSDL: After 15 seconds of real time.
FIIS: After 15 seconds of real time.

9) Identification of Undetected Faults

Bendix: Faults were not individually identified. Although several software programs were executed, in addition to the FCS, it was never determined what proportion of faults, undetected by one program, would be detected by another.
CSDL: No information was given.
FIIS: Faults were individually identified. A fault, which was undetected while executing the inner-loops, was successively injected while executing each outer-loop program. Thus, it was possible to determine the latency of the same fault for different software programs.

10) Length of FCS Program

Bendix: 2200 assembly language instructions.
CSDL: Data not given.
FIIS: 11,000 assembly language instructions.

11) Length of Self-Test Program

Bendix: 346 assembly language instructions
CSDL: Data not given.
FIIS: Approximately 500 assembly language instructions.

Test Results

1) FCS Detection Coverage of Stuck-At Faults

Bendix: 86.5% of gate-level faults (excluding memory bit-faults) were detected, based on 148 injected "care" faults. Only 9% of memory bit-faults were detected.

CSDL:	3637 out of 21,055 faults were undetected (17.3%). Of these, approximately 3000 were definitely identified as "don't cares." Assuming, conservatively, that the remaining 637 were "care" faults, then pin-level coverage was 97%. Of the detected faults, between 2% and 4% required background self-test for their detection.
FIIS:	97% were detected in the open-loop scenario and 98%, in the combined open-loop and closed-loop scenarios, based on 2549 injected, "care" faults.

2) FCS Detection Coverage of Invert Faults

Bendix:	Invert faults were not simulated.
CSDL:	Invert faults were not simulated.
FIIS:	Unweighted, open-loop faults: 1.71% undetected inverts versus 4.37% stuck-ats.

3) FCS Detection Coverage of 2901A Faults

Bendix:	88% of gate-level faults were detected, based on 52 injected "care" faults.
CSDL:	Data not given.
FIIS:	100% detection, based on 423 injected "care" faults.

4) Time to Detect Faults

Bendix:	Most detected faults were detected within 800ms of their occurrence.
CSDL:	Most detected faults were detected within 600ms of their occurrence.
FIIS:	Most detected faults were detected within 500ms of their occurrence; a few faults required up to 6.5 seconds for detection.

5) Self-Test Coverage

Bendix:	97.4% of gate-level faults (excluding memory bit-faults) detected, based on 2234 injected "care" faults. 92% of all gate-level faults were detected, based on 2534 injected "care" faults. 97.6% of pin-level faults were detected, based on 376 injected "care" faults.
CSDL:	Data not given.
FIIS:	100% detected, based on 1687 injected "care" faults.

6) "Don't Care" Faults

Bendix:	15.5% of all gate-level faults were "don't cares". When memory bits were excluded, 7.2% of gate-level faults were "don't cares". 6% of pin-level were "don't cares".
CSDL:	Possibly as much as 17.3% of pin-level faults were "don't cares".
FIIS:	6.11% of pin-level faults were "don't cares".

THIS PAGE LEFT INTENTIONALLY BLANK

9.0 SUMMARY AND CONCLUSIONS OF FIIS EXPERIMENTS

When FIIS results are cited, they refer to unweighted faults with invert faults excluded.

9.1 FAULT DETECTION

1) Most detected faults are detected within a few computational frames of their occurrence, e.g., 10 frames = 500ms. [This corroborates the results of previous studies, e.g., Bendix and Draper.]

2) Faults are activated (i.e., produce errors) primarily by operating software and not by varying sensor inputs, as evidenced by the high coverage in the open-loop scenario.

3) Most faults are activated by the baseline software (96.3%) [these are alpha-faults in the single-fault model. This coverage is considerably better than the Bendix study indicated, which is not surprising considering that the Bendix FCS program was 2200 words whereas the open-loop program was 11,000 words.]

4) A small proportion of open-loop detected faults are detected between 1 and 6.5 seconds (0.3%). [These are alpha-faults with long latency times in the single-fault model.]

5) A small proportion of faults are not activated by the baseline program but are activated by an auxiliary program (1.3%). [These are beta-faults in the single-fault model. This important observation was made possible because a) individual faults were identified in the experiments and b) auxiliary software programs were part of the operating software. This observation is important in constructing a single-fault model.]

6) A small proportion of faults are not activated by either the baseline or auxiliary programs (2.4%). [These faults could accumulate and, if eventually activated, could result in a near-simultaneous loss of lanes. Preflight or background self-test is an effective way to detect these faults.]

7) Faults which are activated by the baseline program produce errors at a somewhat higher rate than faults activated by autopilot modes (and not activated by the baseline program).

8) Detection statistics for weighted and unweighted faults are similar, e.g., 96.51%, open-loop, weighted versus 95.63%, open-loop, unweighted. [A surprising result since the failure rate/pin differs widely over the devices of the caps-6 computer.]

9) Invert faults are less latent than stuck-ats, e.g., for unweighted, open-loop faults, 1.71% undetected inverts versus 4.37% undetected stuck-ats.

10) Micromemory and 2901a faults are 100% detected. [Not surprising since pin-level stuck-at faults on these devices are not very latent.]

11) Detection of input versus output pin faults is approximately the same. [It was expected that input pin faults would be more latent than output pin faults. However, since output faults are, effectively, input faults of downstream devices, the difference in latency is probably not significant when tandem devices are involved.]

12) Self test coverage is 100%. [Impressive, even at the pin-level, especially for a 500 word program. The self test designer was obviously well-acquainted with the hardware.]

13) 6.11% of all faults were "don't cares." [This is consistent with Bendix's results, which estimated 7% at the pin-level. The relatively large number of "don't cares" and the uncertainty and difficulty in identifying them precludes accurate estimates of detection coverage, i.e., the statistical uncertainty is at least several percent.]

14) FIIS test results generally corroborate the results of the Bendix and CSDL studies. This is surprizing considering the dissimilarity of input sensor selection, computer architecture and operational software between the three studies.

15) The validity of the FIIS approach hinges on the validity of pin-level fault models. Until this issue is resolved the results of the FIIS and previous experiments must be considered tenatative. However, the results can be used as a relative measure of fault detection capability.

16) The FIIS results provide a good basis for the construction of a single-fault model, e.g., identification of key states and order-of-magnitude transition rates. This is the most significant output of the experiments. The next logical step is the construction of multiple-fault models and reliability parameter sensitivity studies.

17) The FIIS approach to fault latency and coverage estimation is relatively inexpensive compared with a software simulation approach. For example, it required only 4 man weeks to perform the FIIS experiments. This assumes that the FIIS fault injection hardware and recording equipment is already in place.

18) The FIIS methodology was thoroughly tested during the study. FIIS performance was impressive, especially in the following areas:

a) Fault Generation Capability

Although only S-a-0, S-a-1 and invert faults were injected, FIIS provided the capability of simulating a wide variety of fault types, under software control.

b) Ease of Use

Injecting faults and recording data was tedious but relatively simple after the initial set-up.

c) Fidelity

Since faults were injected into real hardware there was never any question about the fidelity of the experiments nor was there a need to validate the simulation.

d) Speed

All of the FIIS experiments were run in real time. As a consequence, it was possible and practicable to determine detection coverage over long periods of time and while executing a variety of software programs.

9.2 Single-Fault Model Assessment

The results of the FIIS experiments confirmed the structure of the single-fault model, at least to the level of detail it was intended to represent. It was definitely confirmed that the transitions from active fault states are extremely fast relative to the rate of failure occurrences. As a consequence, these rates can be approximated by constants, as shown in (Reference 11). If, in addition, the transitions between benign and active states can also be approximated by constants, then a markov model results. This greatly simplifies the single-fault model.

On the basis of results from the Bendix, CSDL and FIIS experiments we can obtain order-of-magnitude estimates of the parameters of the single-fault model.

The parameter, p1

p1 = proportion of alpha-faults.

From Table 10, $.9625 < p1 < .9783$.
From Bendix, $.85 < p1$.

The Parameter, p2

p2 = proportion of beta-faults active at their occurrence.

This parameter cannot be determined from fault injection experiments. A conservative estimate would be p2=0.

The Parameter, p3

p3 = proportion of beta-faults benign at their occurrence.

Since p3=1-p1-p2 and p2=0, we obtain

1) $.0216 < p3 < .0373$ (FIIS) and
2) $p3 < .15$ (Bendix).

The Parameters, $e\alpha$, $e\beta$

The histograms show that most alpha-faults are detected within 500 ms of their occurrence and that active beta-faults tend to produce errors over longer time intervals, e.g., up to 4 sec. Thus,

$e\alpha > 7200$/hour
$e\beta > 900$/hour

The Parameter, q

q = proportion of auxiliary programs that activate a benign fault.

This parameter could not be obtained since the auxiliary programs used in the experiments were only a subset of the autopilot modes. However, preliminary (and conservative) estimates of survivability can be obtained by assuming that there is only one auxiliary software program, in which case q = 1.

The Parameter, as

as = rate at which auxiliary programs are brought on-line.

This parameter is determined by the operational scenarios of the FCS and cannot be determined by fault injection experiments. The estimated range of as is .01/hour<as<10/hour.

The Parameter, 1/ds

1/ds = average duration of an auxiliary program.

This parameter is also determined by the operational scenarios of the FCS. The estimated range of 1/ds is .01 hour<1/ds<1 hour.

The Parameter, λ

λ - failure rate of an FCS lane.

The current range of λ values is $.001/\text{hour} < \lambda < .0001/\text{hour}$.

9.3 PROBLEM AREAS

The only serious problem encountered during the experiments was the extreme sensitivity of some devices to being placed on the extender board. The associated module of such a device would not function correctly in the presence of an intervening pair of field effect transistors (FETS) and would immediately trigger a comparator alarm. No data was acquired on these sensitive pins. The following devices would not operate correctly on the extender board:

54LS74	D Flip Flop
54LS174	D Flip Flop
54LS175	D Flip Flop
54LS377	D Flip Flop
54LS161	Bit Counter
54LS194	Shift Register

THIS PAGE LEFT INTENTIONALLY BLANK

10.0 RECOMMENDED FUTURE STUDIES

Although the FIIS, Bendix and CSDL studies have contributed significantly to our understanding of fault, error and detection dynamics, a number of important issues connected with FCS survivability assessment remain unresolved. These issues have already been identified in Section 4.2. In order to resolve some of these issues the following studies are recommended:

1. Failure Modes of Digital Devices

   Previous fault injection experiments have employed stuck-at faults either on device pins or internal gate nodes. It remains to be determined to what extent stuck-at faults represent failure modes of real devices (Reference 8). A determination of real failure mode data is required.

2. Pin-Level Versus Gate-Level Faults

   Until actual failure mode data becomes available, experimenters will, no doubt, continue to inject pin-level and gate-level stuck-at faults. The Bendix study indicated a significant difference between pin-level and gate-level detection coverage. At issue is the validity of either fault type and the extent to which one is more or less latent than the other.

3. Single-Fault Model Incorporating Intermittent/Transient Faults

   The strawman single-fault model of Section 4.3 appears to be adequate for permanent faults. The model should be extended to include intermittent/transient faults. Estimates of occurrence, duration and reoccurrence rates are required.

4. Multiple-Fault Models

   In order to perform a reliability assessment of an FCS it is essential to model multiple fault effects. The rudimentary multiple-fault model of Figure 2 is a good starting point.

5. Reliability/Survivability Assessment of an FCS

   The FIIS, Bendix and CSDL studies have provided a database of fault, error and detection dynamics, at least for permanent faults. It now remains to use this data (augmented, if possible, by intermittent/transient fault data) to assess the reliabilty/survivability of an FCS. A reliability parameter sensitivity study is needed in order to a) identify critical ranges of single and multiple-fault model parameters and b) determine inflight, preflight and maintenance test coverage required to reduce latent faults to acceptable levels.

THIS PAGE LEFT INTENTIONALLY BLANK

11.0 REFERENCES

1. Larsen, W. E. and A. Carro, "Digital Avionics Systems - Overview of FAA/NASA/Industry-Wide Briefing," 7th Digital Avionics Systems Conference Proc., October 1986.

2. Mulcare, D. B., J. W. Benson, D. Eldredge, W. E. Larsen, et al., "Hardware Fault Insertion and Instrumentation System (FIIS) Definition Study," DOT/FAA/CT-83-32, FAA Technical Center, June 1983.

3. Benson, J. W., W. E. Larsen, and R. Taper, "Hardware Fault Insertion and Instrumentation System: Mechanization and Validation," Draft DOT/FAA/CT-86-31, November 1986.

4. Lala, J., and T. B. Smith, "Development and Evaluation of a Fault-Tolerant Multiprocessor (FTMP) Computer - Volume III, FTMP Test and Evaluation," NASA CR 166073, NASA Langley Research Center, May 1983.

5. McGough, J. G., F. Swern, "Measurement of Fault Latency in a Digital Avionic Processor," Part II, NASA CR 3651, NASA Langley Research Center, January 1983.

6. Lala, J., and T. B. Smith, "Development and Evaluation of a Fault-Tolerant Multiprocessor (FTMP) Computer - Volume I, FTMP Principles of Operation)," NASA CR 166071, NASA Langley Research Center, May 1983.

7. Lala, J., and T. B. Smith, "Development and Evaluation of a Fault-Tolerant Multiprocessor (FTMP) Computer - Volume II, FTMP Software," NASA CR 166072, NASA Langley Research Center, May 1983.

8. McGough, J. G., "Feasibility Study for a Generalized Gate Logic Software Simulator," NASA CR 172159, NASA Langley Research Center, July 1983.

9. Hopkins, A., T. B. Smith, J. Lala, "FTMP - A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft," Proc. IEEE, Vol. 66, October 1978.

10. McGough, J. G., "Effects of Near-Coincident Faults in Multiprocessor Systems," Proc. of 5th Digital Avionics Systems Conference, November 1983.

11. McGough, J. G., Trivedi, K., Smotherman, M., "The Conservativeness of Reliability Estimates Based on Instantaneous Coverage," IEEE Trans. Computers, July, 1985.

THIS PAGE LEFT INTENTIONALLY BLANK

APPENDIX A

CAPS 6 Components

* FR = Failure Rate, PPMH; Mil 217D, Airborne, Inhabited Transport

I. CAPS 6 Data Path Card

Device	Component	Pins/ Device	Total Pins	FR/ Device	Total FR
Processor (ALU)					
2901A	U14,U15,U17,U18	40	160	.5792	2.3168
Program Counter LSB Mux					
54LS253	U5	16	16	.1177	.1177
Carry Input Sel. Mux					
54LS253	U8	16	16	.1177	.1177
Processor Address Sel. Mux					
54LS253	U9,U10	16	32	.1177	.2354
54LS257	U11	16	16	.0886	.0886
Shift/Rotate Mux					
54LS253	U2,U3	16	32	.1177	.2354
Data Sel. Mux					
54LS253	U21,U22,U23,U28	16	64	.1177	.4708
54LS257	U29.U36	16	32	.0886	.1772
Status Register					
54LS377	U12	20	20	.4258	.4258
Instruction Register					
54LS377	U19,U20	20	40	.4258	.8516
Instruction Syllable Sel. Mux					
54LS257	U30,U35	16	32	.0886	.1772
Address Select Mux					
54LS257	U32,U33,U34,U37	16	64	.0886	.3544
Stack Vector Register					
54LS194	U4	16	16	.1162	.1162

Device	Component	Pins/ Device	Total Pins	FR/ Device	Total FR
Look-Ahead Carry 54LS182	U13	16	16	.0932	.0932
Loop Counter 54LS163	U31	16	16	.1249	.1249
Data Bus Transceiver 7835	U38,U39,U40,U41	16	64	.1892	.7568
Interrupt Controller 2914	U16	40	40	.5792	.5792

Total FR = 7.24

II. CAPS 6 Control Card

Device	Component	Pins/ Device	Total Pins	FR/ Device	Total FR
Instruction Mapper Prom					
HM7643	U1,U7,U13	16	48	.1899	.5697
Control Store Memory					
HM7643	U3,U4,U5,U6,U9, U10,U11,U12,U17, U18	16	160	.1899	1.899
Next Address Control Prom					
HM7603	U8	16	16	.1014	.1014
Transfer Bus Address Register					
54LS377	U40,U42	20	40	.4258	.8516
Control Registers					
54LS377	U24,U25	20	40	.4258	.8516
54LS174	U21,U23	16	32	.3117	.6234
29LS18	U30,U31,U32	16	48	.1024	.3072
Transfer Bus Access Control Registers					
54LS175	U22	16	16	.2983	.2983
Interrupt Decoder					
54LS138	U33	16	16	.2870	.2870
Control Register Decoder					
54LS138	U46	16	16	.2870	.2870
Microprogram Sequencer					
2911	U14,U15,U16	20	60	.4258	1.2774
Clock/Control Logic Register					
54LS175	U34	16	16	.2983	.2983
Clock/Control Logic D FFs					
54LS74	U20A,U20B,U27B U29A,U39B	8	40	.0519	.2595
Transfer Bus Acquisition Logic D FFs					
54LS74	U29B,U39A	8	16	.0519	.1038
Transfer Bus Address Transceiver					
7835	U47,U48,U49,U50	16	64	.1892	.7568
				Total FR =	8.772

THIS PAGE LEFT INTENTIONALLY BLANK

APPENDIX B

Fault Weighting

In the computation of detection coverage each fault was weighted in proportion to the failure rate of its associated device. Thus, pins of devices with large failure rates were given greater weight than pins of devices with smaller failure rates. The relative weighting of each pin is given in the following table. The FR/Pin was obtained by dividing the failure rate of the device by the number of pins.

Device	FR/Pin	Number of Equivalent Faults/Pin
2911	.0213	213
54LS377	.0213	213
54LS174	.0195	195
54LS175	.0186	186
54LS138	.0179	179
2901A	.01448	145
HM7643	.01187	119
7835	.01183	118
54LS163	.00781	78
54LS253	.00736	74
54LS194	.00726	73
54LS74	.0065	65
29LS18	.0064	64
HM7603	.00634	63
54LS182	.00583	58
54LS257	.00554	55

When constructing latency histograms or when estimating coverage, each fault was assumed to be representative of a larger set, the number being equal to the "number of equivalent faults/pin." Thus, for example, if a pin fault of 2911 is detected, it will be counted as though 213 faults were detected (and injected).

END
DATE
FILMED
APRIL
1988
DTIC